Abbas Abdolmaleki Information theoretic stochastic search

UMinho | 2017

**Universidade do Minho**
Escola de Engenharia

Abbas Abdolmaleki

# Information theoretic stochastic search

**The MAP-i Doctoral Programme in Informatics, of the Universities of Minho, Aveiro and Porto**

universidade de aveiro

Universidade do Minho

U.PORTO

October 2017

**Universidade do Minho**
Escola de Engenharia

Abbas Abdolmaleki

# Information theoretic stochastic search

**The MAP-i Doctoral Programme in Informatics, of the Universities of Minho, Aveiro and Porto**



Universidade do Minho

supervisors:
**Professor Doutor Luis Paulo Reis**
**Professor Doutor Nuno Lau**

October 2017

STATEMENT OF INTEGRITY

I hereby declare having conducted my thesis with integrity. I confirm that I have not used plagiarism or any form of falsification of results in the process of the thesis elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, 31/10/2017

Full name: Abbas Abdolmaleki

Signature:

Abbas Abdolmaleki

# *Acknowledgements*

Firstly, I would like to express my sincere gratitude to my advisors Prof. Nuno Lau and Prof. Luis Paulo Reis for the continuous support of my Ph.D study and related research. I could not have imagined having more supportive supervisors for my Ph.D study.

Besides my advisor, I would like to thank Prof. Gerhard Neumann for his support of my research. For his motivation, and immense knowledge. For the sleepless nights we were working together before deadlines. His guidance helped me in all the time of this research.

My sincere thanks also goes to Prof. Jan Peters who provided me an opportunity to join their team as visiting researcher, and who gave access to the laboratory and research facilities. Without that precious support it would not be possible to conduct this research.

I thank my fellow teammates in FC-Portugal robocup team. For the stimulating discussions and for all the fun we have had in the past years.

Last but not the least, I would like to thank my family: my parents, my brothers and sisters for supporting me spiritually throughout this long path and my life in general.

I also gratefully acknowledge the funding received towards my PhD from FCT - Fundação para a Ciência e a Tecnologia. As well as fundings by European Union's FP7 under EuRoC grant agreement CP-IP 608849 and by LIACC (UID/CEC/00027/2015) and IEETA (UID/CEC/00127/2015).

The MAP-I Doctoral Program

# *Abstract*

University of Porto/Aveiro/Minho

Doctor of Philosophy

**Information Theoretic Stochastic Search**

by Abbas Abdolmaleki

Optimization is the research field that studies the design of algorithms for finding the best solutions to problems we may throw at them. While the whole domain is practically important, the present thesis will focus on the subfield of continuous black-box optimization, presenting a collection of novel, state-of-the-art algorithms for solving problems in that class. In this thesis, we introduce two novel general-purpose stochastic search algorithms for black box optimisation. Stochastic search algorithms aim at repeating the type of mutations that led to fittest search points in a population. We can model those mutations by a stochastic distribution. Typically the stochastic distribution is modelled as a multivariate Gaussian distribution. The key idea is to iteratively change the parameters of the distribution towards higher expected fitness. However we leverage information theoretic trust regions and limit the change of the new distribution. We show how plain maximisation of the fitness expectation without bounding the change of the distribution is destined to fail because of overfitting and the results in premature convergence. Being derived from first principles, the proposed methods can be elegantly extended to contextual learning setting which allows for learning context dependent stochastic distributions that generates optimal individuals for a given context, i.e, instead of learning one task at a time, we can learn multiple related tasks at once. However, the search distribution typically uses a parametric model using some hand-defined context features. Finding good context features is a challenging task, and hence, non-parametric methods are often preferred over their parametric counter-parts. Therefore, we further propose a non-parametric contextual stochastic search algorithm that can learn a non-parametric search distribution for multiple tasks simultaneously.

The MAP-I Doctoral Program

# *Abstract*

University of Porto/Aveiro/Minho

Doctor of Philosophy

**Information Theoretic Stochastic Search**

by Abbas Abdolmaleki

Otimização é área de investigação que estuda o projeto de algoritmos para encontrar as melhores soluções, tendo em conta um conjunto de critérios, para problemas complexos. Embora todo o domínio de otimização tenha grande importância, este trabalho está focado no subcampo da otimização contínua de caixa preta, apresentando uma coleção de novos algoritmos novos de última geração para resolver problemas nessa classe. Nesta tese, apresentamos dois novos algoritmos de pesquisa estocástica de propósito geral para otimização de caixa preta. Os algoritmos de pesquisa estocástica visam repetir o tipo de mutações que levaram aos melhores pontos de pesquisa numa população. Podemos modelar essas mutações por meio de uma distribuição estocástica e, tipicamente, a distribuição estocástica é modelada como uma distribuição Gaussiana multivariada. A ideia chave é mudar iterativamente os parâmetros da distribuição incrementando a avaliação. No entanto, alavancamos as regiões de confiança teóricas de informação e limitamos a mudança de distribuição. Deste modo, demonstra-se como a maximização simples da expectativa de "fitness", sem limites da mudança da distribuição, está destinada a falhar devido ao "overfitness" e à convergência prematura resultantes. Sendo derivado dos primeiros princípios, as abordagens propostas podem ser ampliadas, de forma elegante, para a configuração de aprendizagem contextual que permite a aprendizagem de distribuições estocásticas dependentes do contexto que geram os indivíduos ideais para um determinado contexto. No entanto, a distribuição de pesquisa geralmente usa um modelo paramétrico linear em algumas das características contextuais definidas manualmente. Encontrar uma contextos bem definidos é uma tarefa desafiadora e, portanto, os métodos não paramétricos são frequentemente preferidos em relação às seus semelhantes paramétricos. Portanto, propomos um algoritmo não paramétrico de pesquisa estocástica contextual que possa aprender uma distribuição de pesquisa não-paramétrica para várias tarefas simultaneamente.

# Contents

# Chapter 1

# Introduction

This thesis consists of 8 chapters and each chapter is self contained. However in order to build a foundation for the entire thesis, in this chapter we formally introduce optimisation and its different sub classes and we focus on continuous black box optimisation in particular. We discuss evaluation methods used in this thesis, and review some of the most important state-of-the-art methods. Finally we outline the contribution of each chapter and we present the publications resulting from this research.

## 1.1 Problem Definition

The aim of optimisation is to find the best solution to some type of problem, for example, finding the best parameters of a humanoid robot walk controller that leads to highest speed. Formally, function optimisation is defined as finding the best solution $\boldsymbol{\theta}_*$ in a search space $S$ (or solution space), which satisfies:

$$\boldsymbol{\theta}_* = \underset{\boldsymbol{\theta} \in S}{\operatorname{argmax}}\, f(\boldsymbol{\theta})$$

, where $f : S \mapsto \mathbb{R}$ is called the objective function (also known as fitness function). In robot walking example, the search point $\boldsymbol{\theta}$ would be the parameters of the robot walking parameters and $f$ would be speed of the robot achieved using particular parameters. The solution $\boldsymbol{\theta}_*$ does not need to be unique in S, it is sufficient to find one of them though.

### 1.1.1 Continuous Optimisation

Continuous optimisation is the class of optimization problems where $S$ is a subset of $\mathbb{R}^n$ where n is the dimension of the search space. Note that in continuous optimisation while the elements of the search point $\boldsymbol{\theta}$ must be continuous, the $f$ does not need

to be continuous. Working in a continuous search space has both advantages and disadvantages. The problem is that the search space is infinite and therefore in general finding $\boldsymbol{\theta}_*$ in infinite time is not guaranteed. However many real world objective functions are smooth, which allows us to use approaches that take advantages of of smoothness such as gradient ascent that can not be applied on discrete search spaces, therefore in practice, continuous optimisation problems are often easier to solve than the discrete ones that usually are NP-hard. The subfield complementary to continuous optimization is discrete optimization, and includes, among others, integer programming. Hybrid cases exist as well, where part of the solution is continuous and part is discrete, e.g., mixed integer programming problems. A constrained continuous search space is a subset of $\mathbb{R}^n$ bounded by a collection of inequalities (generally linear ones), which define the set of feasible solutions. Constraints add additional difficulty to optimization problems, in particular when the optimum lies on the search space boundary (i.e., is not an interior point). In this case, optimization methods require an initial feasible solution $\boldsymbol{\theta}_{initial}$ from which to start the optimization. Entire research fields are dealing with the issues specific to constrained optimization (most notably,convex optimization), but approaches designed for unconstrained optimization can be applied too: for example, the objective function can be modified to include a penalty term that keeps the solutions found within the feasible region, after which continuous optimization techniques can be employed to solve the constrained problem.

### 1.1.2 Black-Box Optimisation

While optimization was originally developed for objective functions that were known, but where the optimum could not be calculated analytically, some of the algorithms apply equally well to the case where the function is unknown, which spawned the subfield of black-box optimization (highlighting the fact that the objective function is like a black box, we know nothing about its internals, can only observe what goes in, and what comes out). The field is also known as direct search, or derivative-free optimization, because one of the key unavailable pieces of information is the derivative of the function, which many other optimization methods rely on; it is closely related to the field of metaheuristics. Also, most algorithms for black-box optimization fall into the category of randomized search methods, which introduce randomness in the search process. It is worth noting that while the methods of black-box optimization were developed for problems where the function is unknown, they are sometimes applied successfully when the analytical form is available. Prominent examples are non-differentiable functions, deceptive functions (where gradient information might

lead the search astray) and functions with uninformative gradients (e.g., piece-wise constant).

### 1.1.3 Local Optimisation

Many continuous optimization problems are multi-modal, that is, there exist multiple solutions $\boldsymbol{\theta}$ that are locally optimal. In general, only one local optimum is also the global optimum. It is challenging to determine whether the best local optimum found so far is $\boldsymbol{\theta}_*$, but for some purposes a local optimum is a sufficient result. We thus distinguish global optimization methods, which aim at finding the true optimum $\boldsymbol{\theta}_*$, and local optimization methods which contend themselves with a (reasonably good) local optimum. A local optimization algorithm with a randomized component can be restarted multiple times, in order to increase the probability of hitting the global optimum, or just reaching a better local optimum.

### 1.1.4 Optimizing noisy functions

In many real-world problems, it is unrealistic to assume that we can measure the objective function $f$ precisely, as it is easily corrupted by measurement or process noise, and multiple measurements of $f(\boldsymbol{\theta})$ do not give the same result for a given $\boldsymbol{\theta}$. Some black-box optimization methods are ill-equipped to deal with noise (see Auger et al., 2010), which underlines the need for robust approaches.

### 1.1.5 Optimising Contextual Objective Functions

So far we assumed the objective function is fixed and the solution is a point estimate $\boldsymbol{\theta}^*$ in the search space. Now consider an optimisation task where the objective function changes slightly dependent on the context of the task. For example in our robot walking task assume we are interested in finding optimal controller parameters for any feasible speed $\boldsymbol{s}$. In this case the solution is not only a point $\boldsymbol{\theta}$ but a function of context $\boldsymbol{s}$, i.e, $m(\boldsymbol{s})$. More formally we consider contextual optimization problems where the objective function depends on a $n_s$-dimensional context vector $\boldsymbol{s}$. Now the optimisation task is to find for each context vector $\boldsymbol{s}$, an optimal parameter vector $\boldsymbol{\theta}^*_{\boldsymbol{s}}$ that maximizes an objective function $\boldsymbol{f}(\boldsymbol{s}, \boldsymbol{\theta}) : \{\mathbb{R}^{n_s} \times \mathbb{R}^{n_\theta}\}. \rightarrow \mathbb{R}$. However in continuous context space we have infinite context vectors therefore in this case we want to find an optimal context dependent policy $m^*(\boldsymbol{s})$ that can generalize the solutions for different contexts.

### 1.1.6 The problems studied in this thesis

For the remainder of this thesis we only consider unconstrained, continuous black box optimisation algorithms for non-contextual, contextual, noise free and noisy problems.

## 1.2 Evaluation criteria

To evaluate and compare different optimization algorithms o n the same problem, we have basically three criteria that the algorithm should minimize:

1. The number of function evaluations until the optimum is found.
2. the total computation time until the optimum is found,
3. the cumulative regret after evaluating candidate solutions $\boldsymbol{\theta}^1$ to $\boldsymbol{\theta}^n$, which is the difference between the sum of objective values and the optimum had been known in advance.

For some applications, some of these measures will be more useful than others. In this thesis we will focus on the first measure, because it is the simplest one, and independent of implementation and hardware questions (unlike the second). Also, counting function evaluations puts the emphasis on the final outcome of the optimization. Additionally we are interested in optimising robotics task where small reduction in the required number of evaluations justifies a significant investment of computational resources. Clearly, evaluating and comparing algorithms on a single problem is not sufficient to determine their quality, as much of their benefits in their performance generalizing to large classes of problems. One of the goals of research in optimization is, arguably, to provide practitioners with reliable, powerful and general-purpose algorithms. This is why we test our algorithms on a whole battery of benchmark functions, taken from different problem classes including robotics.

## 1.3 State-of-The-Art Black Box Optimisation Algorithms

Here, we will briefly review the spectrum of methods that have been applied to continuous, unconstrained black-box optimization (some of which are more generally applicable). Attempting to be exhaustive would be very difficult given the breadth of the field. Instead we will try to point out the most representative algorithms, going into the most depth for those approaches that are currently producing state-of-the-art results, and that are most closely related to our own. A first class of methods

was inspired by classic optimization methods, including simplex methods such as Nelder-Mead [71], as well as members of the quasi-Newton family of algorithms [25]. Simulated annealing [49], a popular method introduced in 1983, was inspired by thermodynamics, and is in fact an adaptation of the Metropolis-Hastings algorithm [36][64].

### 1.3.1 Evolutionary and Stochastic Search Methods

Most prominent among the more heuristic methods are those inspired by evolution, developed from the early 1950s on, including the broad class of genetic algorithms [28] [37]. Mimicking natural evolution [18], these approaches maintain a population of individuals or search points that are evaluated in batch. In each generation the best individuals i.e., the ones with the highest objective value $f$ are selected to survive and produce offspring, i.e., the population in next generation, while other are discarded. Producing can introduce small-scale changes to the search points known as mutation, or perform large scale recombinations of search points. Among the most successful GA for continuous optimisation are differential evolution [89] where the mutations are based upon the difference vectors between the members of the population, and the related particle swarm optimisation[48]. Estimation of distribution algorithms (EDA; [56]) on the other hand rely on modelling the population of (fittest) search points by a distribution, from which new individuals are then drawn; in contrast to GA, using only the estimated distribution and not the old population. Among its representatives are Estimation of Multivariate Normal Algorithm (EMNA) and the closely related cross-entropy method (CEM; [82]), where the search distribution is a multivariate Gaussian, as well as Fitness Expectation Maximization (FEM; [99]) where the distribution is updated using an expectation-maximization approach. Evolution strategies (ES), introduced by Ingo Rechenberg and Hans-Paul Schwefel in the 1960s and 1970s ([80]; [87]), were designed to cope with high-dimensional continuous-valued domains and have remained an active field of research for more than four decades ([9]). They are distinct from GA in that mutations modify each of the continuous decision variables simultaneously, but only very slightly; this process, after several generations, was shown to lead to reasonable to excellent results for many difficult continuous optimization problems. The algorithm framework has been developed extensively over the years to include self-adaptation of the search parameters, and the representation of correlated mutations by the use of a full covariance matrix. This allowed the framework to capture interrelated dependencies by exploiting the covariances while ‘mutating’ individuals for the next generation. The culminating algorithm, covariance matrix adaptation evolution strategies (CMA-ES; [33]), has proven successful in numerous studies (e.g., [26];[70];[60]). We call all

these methods stochastic search methods in the sense that these methods use stochastic mutations in order to explore the search space.

### 1.3.2 Policy Search Methods

Policy search methods [20] has been developed within the Reinforcement Learning community. The goal of RL [93] is to optimize the behavior of an agent that interacts with an environment without being told the correct behavior (as in supervised learning); instead, the agent is only informed about how well it did, in terms of a scalar value called reward. There is a double link between RL and optimization. On one hand, we may consider optimization to be a simple sub-problem of RL, with only a single environment state and a single time-step per episode, where the fitness corresponds directly to the reward (i.e., a bandit problem). On the other hand, more interestingly, the return of a whole RL episode can be interpreted as a single fitness evaluation, where the search space is the policy parameters. In this case, policy search methods [20] in RL are equivalent to black-box optimization. Moreover, when the exploration in parameter space is normally distributed, a direct link between RL and stochastic search methods such as evolution strategies can be established.

### 1.3.3 Response Surface Methods

The standard tool for global optimization are response surface methods (RSM; [11];[68]). They store all available evaluations (some possibly given in advance) and use them to model the cost function, which useful for dimensionality reduction, visualization, assessing certainty, and ultimately determining good points to explore ([10];[45]). A multitude of regression techniques have been used for modeling the response surface, from the original polynomials ([11]) to more recent Gaussian processes ([44];[79]). In addition, a statistical model of the cost function allows expert knowledge to be incorporated in the form of a Bayesian prior.

### 1.3.4 Applications of black box optimisation

Many real world optimization problems that are too difficult or complex to model directly have been solved in using black-box optimization. In [84] see an exhaustive list of applications that use CMA-ES as one of the state-of-the-art algorithms. Here, we give a selection of work in different fields that illustrate the very broad applicability of the framework. In robotics, black box optimisation techniques were used to learn robot skills such robot hopping [38] and walking [13]. In health sciences, optimization techniques were employed for matching CT-scans to ultra-sound images

([100]), and for forensic identification ([39]). In chemistry, black-box optimization were used for chromatography ([42]) and finding stable crystalline structures ([95]). In urban development, blackbox methods have helped optimize resource flows ([46]), groundwater quality ([8]), power distribution ([65]; [97]) and radio network design ([63]). Furthermore, optimization techniques can aid in determining appropriate features, for example for speaker identification ([15]). In space research, they have been utilized, among others, for evolving orbit transfer maneuvers ([66]) and docking approaches ([57]). They also have a long tradition in device design (e.g. [50]) or aeronautics ([10]; [35]), as well as control ([34]). Besides the broad range of direct applications, black-box optimization is also a common component of other machine learning techniques. It is used to train neural networks ('neuro-evolution', see e.g. [29] and [85]), to optimize kernel parameters (model selection, [26].

## 1.4 Contributions of the Thesis

In this thesis, we present a collection of novel, state-of-the-art algorithms for black box optimisation. We briefly explain contribution of each chapter as follows.

In chapter 2 we study weighted maximum likelihood estimate (WMLE) policy seacrh methods. Many episode-based (or direct) policy search algorithms, maintain a multivariate Gaussian distribution as search distribution over the parameter space of some objective function. One class of algorithms, such as episodic REPS, PoWER or $PI^2$ uses, a weighted maximum likelihood estimate (WMLE) to update the mean and covariance matrix of this distribution in each iteration. However, due to high dimensionality of covariance matrices and limited number of samples, the WMLE is an unreliable estimator. The use of WMLE leads to over-fitted covariance estimates, and, hence the variance/entropy of the search distribution decreases too quickly, which may cause premature convergence. In order to alleviate this problem, the estimated covariance matrix can be regularized in different ways, for example by using a convex combination of the diagonal covariance estimate and the sample covariance estimate. In chapter two, we propose a new covariance matrix regularization technique for policy search methods that uses the convex combination of the sample covariance matrix and the old covariance matrix used in last iteration. The combination weighting is determined by specifying the desired entropy of the new search distribution. With this mechanism, the entropy of the search distribution can be gradually decreased without damage from the maximum likelihood estimate.

CMA-ES is one of the most popular stochastic search algorithms. It performs favourably in many tasks without the need of extensive parameter tuning. In chapter

3 we will fully derive all CMA-ES update rules within the framework of expectation-maximisation-based stochastic search algorithms using information-geometric trust regions. We show that the use of the trust region results in similar updates to CMA-ES for the *mean* and the *covariance* matrix while it allows for the derivation of an improved update rule for the step-size.

Many stochastic search algorithms are designed to optimize a fixed objective function to learn a task, i.e., if the objective function changes slightly, for example, due to a change in the situation or context of the task, relearning is required to adapt to the new context. In chapter 4, we extend the well known CMA-ES algorithm to the contextual setting and illustrate its performance on several contextual tasks. Our new algorithm, called contextual CMA-ES, leverage from contextual learning while it preserves all the features of standard CMA-ES such as stability, avoidance of premature convergence, step size control and a minimal amount of parameter tuning.

In chapter 5 we introduce a new surrogate-based stochastic search approach. We learn simple, quadratic surrogate models of the objective function. As the quality of such a quadratic approximation is limited, we do not greedily exploit the learned models. The algorithm can be misled by an inaccurate optimum introduced by the surrogate. Instead, we use information theoretic constraints to bound the 'distance' between the new and old data distribution while maximizing the objective function.

In chapter 6 we introduce a novel non-parametric contextual stochastic search algorithms called Local CECER. We will show that local CECER leverages from a fully context dependent policy update and it is able to learn non-linear policies. We showed that local CECER outperforms the other contextual algorithms.

In chapter 7, we consider a soccer 3d Robocup application. The task is learning kicks with different distances. Using the contextual algorithms we designed in the previous chapters we can successfully train a simulated humanoid robot to kick the ball with different distances in one run optimisation.

And finally chapter 8, summarises the thesis and presents some interesting future research directions.

## 1.5 Publications

In this section we outline the papers being published in context of this PhD thesis:

1. Abdolmaleki, A.; Lua, N.; Reis, L.P ; Price, B.; Neumann, G. (2017). Contextual Covariance Matrix Adaptation Evolution Strategy, International Joint Conference on Artificial Intelligence (IJCAI) (25% acceptance rate).

2. Abdolmaleki, A.; Price, B.; Lua, N.; Reis, L.P ; Neumann, G. (2017). Deriving and Improving CMA-ES with Information Geometric Trust Regions, The Genetic and Evolutionary Computation Conference (GECCO) (Best Paper Award Candidate) (32% acceptance rate)

3. Tangkaratt V., Abdolmaleki A.,Sugiyama M.(2017). Deep Reinforcement Learning with Relative Entropy Stochastic Search, arXiv:1705.07606v1

4. Abdolmaleki, A.; Lau, N.; Reis, L.; Neumann, G. (2016). Non-parametric Contextual Stochastic Search, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

5. Abdolmaleki, A.; Lioutikov, R.; Peters, J.; Lau, N.; Reis, L.; Neumann, G. (2015). Model-Based Relative Entropy Stochastic Search, Advances in Neural Information Processing Systems (NIPS), MIT Press (22% acceptance rate).

6. Akrour, R.; Abdolmaleki, A.; Abdulsamad, H.; Neumann, G. (2016). Model-Free Trajectory Optimization for Reinforcement Learning of Motor Skills, Proceedings of the International Conference on Machine Learning (ICML) (24% acceptance rate).

7. Abdolmaleki, A.; Simoes, D.; Lau, N.; Reis, L.; Neumann, G. (2016). Learning a humanoid kick with controlled distance, RoboCup 2016: Robot Soccer World Cup XX. Springer (Oral Presentation , 24% acceptance rate)

8. Abdolmaleki, A. and Lau, N. and Reis, L. and Neumann, G. (2015). Regularized Covariance Estimation for Weighted Maximum Likelihood Policy Search Methods, Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)

9. Abdolmaleki, A.; Lau, N.; Reis, L.; Peters, J.; Neumann, G. (2015). Contextual Policy Search for Generalizing a Parameterized Biped Walking Controller, In Autonomous Robot Systems and Competitions (ICARSC) (Best Paper Award).

10. Abdolmaleki, A.; Lau, N.; Reis, L.; Peters, J.; Neumann, G. (2015). Contextual policy search for linear and nonlinear generalization of a humanoid walking controller, In Journal of Intelligent and Robotic Systems.

11. Abdolmaleki, A.; Peters, J.; Lau, N.; Reis, L.; Neumann, G. (2014). Contextual Policy Search for Learning a Flexible Bipedal Robot Locomotion Controller, In Dynamic Walking Conference.

12. Abdolmaleki, A.; Shafii, N.; Lau, N.; Reis, L.; Peters, J; Neumann, G. (2014). Omnidirectional Walking with a Compliant Inverted Pendulum Model, In IBERAMIA Conference.

13. Shafii, N.; Abdolmaleki, A.; Lau, N.; Reis, L.(2015). Development of an Omnidirectional Walk Engine for Soccer Humanoid Robots, In International Journal of Advanced Robotic Systems.

14. Shafii, N.; Ferreira R.; Abdolmaleki, A.; Lau, N.; Reis, L. (2013). Omnidirectional Walking and Active Balance for Soccer Humanoid Robot, In EPIA Conference.

15. Ferreira R.; Shafii, N.; Lau, N.; Reis, L. ; Abdolmaleki, A. (2013). Diagonal Walk Reference Generator based on Fourier Approximation of ZMP Trajectory, In Autonomous Robot Systems and Competitions Conference (ICARSC).

16. Abdolmaleki A. ; Movahedi M. ; Lau N. ; Reis L. (2013). A distributed cooperative reinforcement learning method for decision making in Fire Brigade Teams, In Robocup Symposium.

17. Abdolmaleki A. ; Movahedi M. ; Salehi S. ; Lau N. ; Reis L. (2011). A reinforcement learning based method for optimizing the process of decision making in fire brigade agents, In EPAI conference.

# Chapter 2

# Regularized Covariance Estimation for Weighted Maximum Likelihood Policy Search Methods

Many episode-based (or direct) policy search algorithms, maintain a multivariate Gaussian distribution as search distribution over the parameter space of some objective function. One class of algorithms, such as episodic REPS, PoWER or $PI^2$ uses, a weighted maximum likelihood estimate (WMLE) to update the mean and covariance matrix of this distribution in each iteration. However, due to high dimensionality of covariance matrices and limited number of samples, the WMLE is an unreliable estimator. The use of WMLE leads to over-fitted covariance estimates, and, hence the variance/entropy of the search distribution decreases too quickly, which may cause premature convergence. In order to alleviate this problem, the estimated covariance matrix can be regularized in different ways, for example by using a convex combination of the diagonal covariance estimate and the sample covariance estimate. In this research, we propose a new covariance matrix regularization technique for policy search methods that uses the convex combination of the sample covariance matrix and the old covariance matrix used in last iteration. The combination weighting is determined by specifying the desired entropy of the new search distribution. With this mechanism, the entropy of the search distribution can be gradually decreased without damage from the maximum likelihood estimate.

## 2.1 Introduction

Stochastic search algorithms are gradient-free black-box optimizers of some objective function $R_{\boldsymbol{\theta}}$ dependent on a high-dimensional parameter vector $\boldsymbol{\theta}$. Stochastic search algorithms do not put any assumption on the structure of the objective function, such as a Markov assumption. In this research, we focus on episode-based

policy search methods in robotics which are a special case of stochastic search methods. Due to its simplicity, episode based policy search is one of the most successful reinforcement learning approaches in robotics [20, 92, 90, 55]. Episode-based policy search methods address the continuous state-action problems in reinforcement learning by directly optimizing the parameters $\boldsymbol{\theta}$ of a control lower-level policy. Fourier series, splines and DMPs[41] has been commonly used as control lower-level policy in robotics. Policy search methods, directly search over the parameter space of the lower-level policy using an upper-level policy or search distribution which is typically implemented as a multivariate Gaussian distribution. Many state of art methods such as episodic REPS [55], CMA-PI$^2$ [90] and PoWER [53] estimate the Gaussian upper level policy (mean and covariance matrix) by a weighted maximum likelihood estimate (WMLE), see Equation 2.4. To do so, they generate samples from the current upper-level policy and use the return of the samples to estimate the quality of the samples. This quality estimate results in a weight for each sample that can be used to estimate a new mean and a new covariance matrix for the new Gaussian upper-level policy by using a WMLE. Yet, due to high dimensionality of a covariance matrix and limited number of samples, the WMLE estimate of the covariance matrix is an unreliable estimator with high variance. This over-fitted estimation of the covariance makes the upper-level policy highly biased to a specific region of the parameter space, which often causes premature convergence [47]. Instead, we can estimate only a diagonal covariance matrix with fewer parameters [81], yet, such a solution has a high bias and might result in a slow learning performance as we neglect the correlations between the parameters. One other solution is using regularization techniques for estimating the covariance matrix. Standard regularization techniques such as covariance shrinkage [88, 47] are based on a convex combination of different estimators, e.g., the high variance estimator of the sample covariance matrix and the high bias estimator of the diagonal covariance matrix. Yet, policy search algorithms have a big advantage when estimating the covariance matrix. They have access to the covariance with which the (unweighted) samples have been generated. Therefore we propose a new regularization technique that combines the sample covariance estimate with the covariance matrix of the generating distribution, i.e., the old upper-level policy, can be used as a prior in our estimation. Furthermore, we know that controlling the exploration rate in policy search is crucial. The variance/entropy of the upper-level policy should decrease slowly in order to give the algorithm enough time to converge to a (local) optimal solution. Hence, the combination factor of the prior (old covariance) and the sample covariance can be determined by an entropy reduction criterion. At each iteration, we want the entropy of the upper-level policy to decrease for a certain amount. We chose the combination factor between the

two matrices such that the entropy of the resulting distribution is exactly at this desired level. We name our method Covariance Estimation with Controlled Entropy Reduction (CECER). Intuitively, our method can be seen as weighted averaging of covariance matrix estimates of all iterations, where the influence of the initial distribution is decreased at each iteration. Similar combinations of old and new covariance matrices have been used by other stochastic search algorithms such as CMA-ES [32]. We compare different covariance estimation techniques including covariance shrinkage [88] to our new regularization technique based on entropy reduction in context of state of art episode-based policy search methods such as REPS [55] and an episode-based version of PI$^2$[90][1]. The resulting episode-based policy search algorithms are also compared to the Natural Evolution Strategy [98] and CMA-ES [32] on two simulated robotics tasks including a planar arm reaching task and a planar arm hole reaching task. Our algorithm performs favorably in our experiments.

## 2.2 Weighted Maximum Likelihood based policy search

We want to maximize an objective function $R(\boldsymbol{\theta})$,

$$R : \mathbb{R}^n \rightarrow \mathbb{R} \quad , \boldsymbol{\theta} \mapsto R(\boldsymbol{\theta}).$$

The goal is to find one or more parameter vectors, $\boldsymbol{\theta} \in \mathbb{R}^n$, with an objective value, $R(\boldsymbol{\theta})$, as big as possible. The only accessible information on $R(\boldsymbol{\theta})$, are function values $\{R^{[k]}\}_{k=1...N}$ of evaluated parameter vectors $\{\boldsymbol{\theta}^{[k]}\}_{k=1...N}$, where $k$ is the index of the sample and $N$ is number of samples. Episode-based policy search algorithms [20, 92, 90] typically maintain an upper-level policy or search distribution $\boldsymbol{\pi}(\boldsymbol{\theta})$, over the parameter space $\boldsymbol{\theta}$ of a parametrized lower-level policy. Typically, the upper-level policy $\pi(\boldsymbol{\theta})$ is implemented as a multivariate Gaussian distribution, i.e., $\pi(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. In each iteration, the upper-level policy $\boldsymbol{\pi}(\boldsymbol{\theta})$ is used to create samples $\boldsymbol{\theta}^{[k]}$ of the parameter vector $\boldsymbol{\theta}$ of the lower-level policy. Subsequently, the return $R^{[k]}$ of $\boldsymbol{\theta}^{[k]}$ is obtained by evaluating the performance of the lower-level policy with the parameter vector $\boldsymbol{\theta}^{[k]}$. Using the samples and their returns $\{\boldsymbol{\theta}^{[k]}, R^{[k]}\}_{k=1...N}$, a new upper-level policy is computed [2] by either computing gradient based updates [92, 83], covariance matrix adaptation updates [32] or weighted MLE-based updates [90, 52, 55, 94]. We are particularly interested in weighted MLE-based policy search methods which have been shown to be able to outperform gradient-based methods

[1] In the episode-based case, PoWER [53] and PI$^2$ [90] are equivalent.

[2] The goal is that, the new upper-level policy or new search distribution spans samples with higher returns than the old upper-level policy

such as Natural Actor-Critic [76]. WMLE-based policy search methods use the return $R^{[k]}$ to compute a weight $w^{[k]}$ for each sample $\boldsymbol{\theta}^{[k]}$ such that $\sum_{k=1}^{N} w^{[k]} = 1$ [3] and, subsequently, the mean and covariance matrix of the upper-level policy $\boldsymbol{\pi}(\boldsymbol{\theta})$ is updated by a weighted MLE (Equation 2.4). The next we will explain how the weights are computed.

### 2.2.1 Computation of the Weighting

The weight $w^{[k]}$ for sample $\boldsymbol{\theta}^{[k]}$ can be estimated by an exponential transformation of the corresponding return $R^{[k]}$, i.e.,

$$w^{[k]} \propto \exp(R^{[k]}/\eta), \tag{2.1}$$

where $\eta$ specifies the temperature of the exponential transformation, such as applied by the PI$^2$ algorithm [94, 90], PoWER [52] and REPS [55]. The next we will explain how different algorithms set the $\eta$.

**PoWER and PI$^2$** In the PI$^2$ and PoWER algorithms, the temperature parameter $\eta$ is chosen by a heuristic. PI$^2$ chooses

$$\eta = \lambda(\max_k R^{[k]} - \min_k R^{[k]}),$$

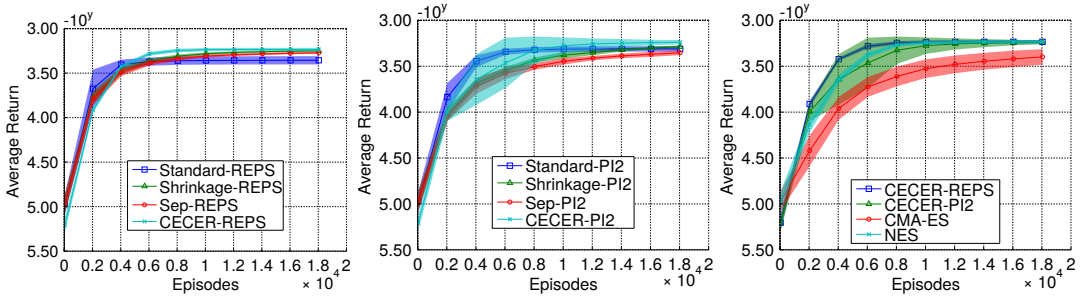
where $R^{[k]}$ is the return of sample $\boldsymbol{\theta}^{[k]}$ and $\lambda$ is typically set between 5 and 15. For PoWER, $\eta$ is often hand tuned. While PoWER and PI$^2$ are actually equivalent if the same strategy for $\eta$ is used[4], both algorithms are derived from very different principles.

**REPS** REPS [75, 55] bounds the Kullback-Leibler divergence between the old policy $q(\boldsymbol{\theta})$ used for sampling and the newly estimated policy $\pi(\boldsymbol{\theta})$. The policy update can hence be formulated as constrained optimization problem where we want to maximize the expected return of the new policy under the KL constraint, i.e.,

$$\begin{aligned} \pi^* = &\text{argmax}_\pi \int \pi(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &\text{s.t. } \text{KL}(\pi(\boldsymbol{\theta}) || q(\boldsymbol{\theta})) \leq \epsilon, \qquad \int \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} = 1 \end{aligned} \tag{2.2}$$

[3] Each weight is a pseudo-probability for the corresponding sample

[4] This is true at least for the episode-based version that neglects the time steps.




(a) REPS (b) $PI^2$ (c) Stochastic Search Algorithms

FIGURE 2.1: The performance comparison for reaching task using a 5 link planar robot. The results show that CECER outperforms the other covariance update methods for the both REPS and $PI^2$. In (c) we see that, CECER-REPS has faster learning rate than the other algorithms.

The main intuition behind this bound is that we can directly control the exploration-exploitation trade-off with the $\epsilon$ parameter. For a large $\epsilon$ (exploitation), the entropy/variance of the new upper level policy will shrink quickly such that, it will always choose the sample with highest return in our dataset while for a small $\epsilon$ (exploration), the new search policy and the old search policy would be almost identical. While this optimization problem can not be solved analytically as $R_{\boldsymbol{\theta}}$ is unknown, it can be solved for our samples $\{\boldsymbol{\theta}^{[k]}, R^{[k]}\}_{k=1\ldots N}$. The solution for the sample based problem results in a weight $w^{[k]} \propto \exp(R^{[k]}/\eta)$ for each sample, where the temperature parameter $\eta$ can be found by optimizing the dual function

$$g(\eta) = \eta\epsilon + \eta \log\left(\sum_{k=1}^{N} \frac{1}{N} \exp\left(\frac{R^{[k]}}{\eta}\right)\right) \tag{2.3}$$

of the optimization problem. The optimal value for $\eta$ can be obtained by minimizing the dual function $g(\eta)$ such that $\eta > 0$, see [55, 12]. The next, we will explain how the weightings can be used to update the upper-level policy.

### 2.2.2 Weighted ML Policy Updates

In each iteration, after computing the weightings $w^{[k]}_{k=1\ldots N}$, the new upper level policy is computed by using the samples and their weightings $\{\boldsymbol{\theta}^{[k]}, w^{[k]}\}_{k=1\ldots N}$. PI$^2$ , PoWER and REPS directly use an unbiased weighted maximum likelihood estimate [20] for estimating $\boldsymbol{\mu}$ and the sample covariance $\boldsymbol{S}$ of a Gaussian upper level policy which is given by

$$\boldsymbol{\mu} = \sum_{i=1}^{N} w^{[k]} \boldsymbol{\theta}^{[k]}, \quad \boldsymbol{S} = \frac{\sum_{i=1}^{N} w^{[k]} (\boldsymbol{\theta}^{[k]} - \boldsymbol{\mu})^T (\boldsymbol{\theta}^{[k]} - \boldsymbol{\mu})}{1 - \sum_{i=1}^{N} \left(w^{[k]}\right)^2}. \tag{2.4}$$

Here, the sample covariance matrix of a $p$ dimensional parameter space has $n = \frac{p+p^2}{2}$ free parameters to estimate. Typically, the number of samples used for the estimate is much smaller than this number of free parameters. In this case, it has been shown that the sample covariance matrix from Equation 2.4 is not a good estimate of the true covariance matrix [88] and biases the search distribution towards a specific region of the search space. Due to this effect, the search distribution uncontrollably looses its exploration/entropy along many dimensions of the parameter space and will therefore causes premature convergence. That is a highly unwanted effect in policy search. Alternatively, instead of estimating a full covariance matrix, we could estimate a diagonal covariance matrix which has fewer parameters to estimate and, hence, will not suffer so severely from over-fitting. However, using a diagonal covariance matrix neglects the correlations between the parameters, which might again lead to a slow learning progress [17, 81].

## 2.3 Covariances Regularization for ML based Policy Search

One way to achieve a more accurate covariance estimate is to use regularization techniques that combine the sample estimate of the covariance matrix with a target estimate of the covariance matrix [88]. Different target covariance estimate can be used such as the diagonal covariance matrix or even an identity matrix that is multiplied with some factor [88]. In policy search, we also have the possibility to use the old covariance matrix as target covariance estimate, as we know that the unweighted samples have been generated using it. There are different ways to determine the interpolation factor between the sample covariance and the target covariance estimate. We will discuss first a standard algorithm for determining this interpolation factor and subsequently present our new method based on a controlled reduction of the entropy of the resulting policy.

### 2.3.1 Combining Diagonal and Full Covariance Matrix Estimates by Covariance Shrinkage

In covariance shrinkage estimation [88], we shrink a high-dimensional estimated covariance $\boldsymbol{S}$ towards a lower-dimensional covariance $\boldsymbol{G}$ with fewer parameters (e.g. diagonal matrix) by a weighted average, i.e.,

$$\boldsymbol{\Sigma} = \lambda \boldsymbol{G} + (1 - \lambda)\boldsymbol{S} \tag{2.5}$$

where $\lambda \in [0, 1]$ is the shrinkage intensity. It has been shown In [88], that the combination of covariance estimators with high bias(e.g. diagonal covariance) and high variance (sample covariance) in Equation 2.5 gives us a regularized estimate that outperforms each of those two estimators in terms of estimation error. In the case of our policy update, the matrix $\boldsymbol{G}$ is a diagonal covariance matrix(with $p$ parameters) and $\boldsymbol{S}$ is a sample covariance matrix (with $\frac{p+p^2}{2}$ parameters) estimated by the samples $\{\boldsymbol{\theta}^{[k]}, w^{[k]}\}_{k=1...N}$. Intuitively, in this method, we want to shrink the overestimated correlations between parameters in matrix $\boldsymbol{S}$ towards zero to get a better conditioned covariance matrix. And the diagonal elements will stay unchanged. To do so, we parametrize our desired covariance matrix for the policy update in terms of variances and correlations, i.e.,

$$\Sigma_{ij} = \begin{cases} S_{ij} & \text{if } i = j, \\ R^*_{ij}\sqrt{S_{ii}S_{jj}} & \text{if } i \neq j, \end{cases} \tag{2.6}$$

where $R^*_{ij}$ is the element of the shrunk correlation matrix, i.e,

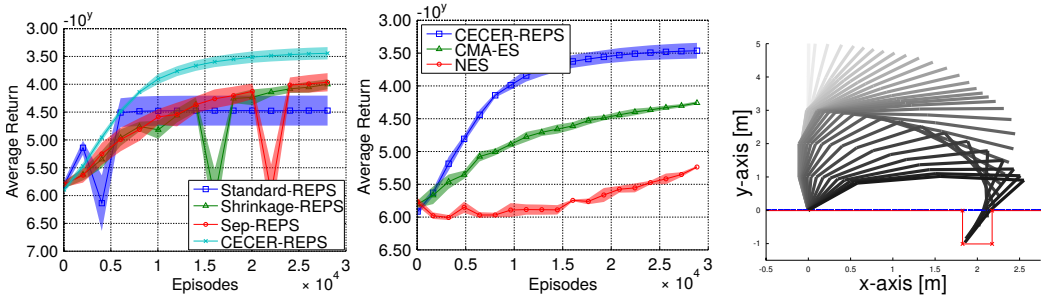
$$R^*_{ij} = \begin{cases} 1 & \text{if } i = j, \\ R_{ij}\min(1, \max(\sigma, 1 - \lambda^*)) & \text{if } i \neq j. \end{cases} \tag{2.7}$$

where $\lambda^*$ is the optimum shrinkage intensity. The min-max term in Equation 2.7 is used for limiting $\lambda^*$ between $0$ and $1 - \sigma$. Typically, $\sigma = 0$ is used. Yet, we empirically found that policy search algorithms performed slightly better if we set $\sigma$ to

$$\sigma = \min\left(\frac{\phi_{\text{eff}}}{p^2}, 1\right), \quad \phi_{\text{eff}} = \frac{1}{\sum_{k=1}^{N}(w^{[k]})^2}, \tag{2.8}$$

where $\phi_{\text{eff}}$ is the number of effective samples which is computed as in [32] and $p$ is the number of dimensions of the parameter vector $\boldsymbol{\theta}$. The reason is that, covariance matrices that need to be estimated for our policy search methods are high dimensional considering the small amount of data that we want to use. As a consequence, matrix shrinkage algorithms will, in many cases, just decide to take the estimator with less variance (which is the diagonal covariance matrix) with a factor of 100%. With this rule we force the shrinkage algorithm to always take a small part from the full sample covariance matrix therefore the algorithm always exploits the correlations between parameters. Typically, $\sigma$ has a very small value close to $0$. Next we will explain how the value of $\lambda^*$ is computed.

**Computing the Shrinkage Intensity** We can find the optimum $\lambda^*$ efficiently in closed form using the method given in [88]. This results in an optimal lambda value




(a) REPS (b) Stochastic Search Algorithms (c) Planar Reaching Task

FIGURE 2.2: The performance comparison for high dimensional reaching task using a 20-link planar robot. The results show that CECER clearly outperforms the other covariance update methods for REPS policy update and CECER-REPS has better performance than NES and CMA-ES (c) The planar hole reaching task used for our comparisons. A 5-link planar robot has to reach the bottom of a hole centring at point [2 0] in task space while avoiding any collision. The hole is indicated by the red lines. The postures of the resulting motion are shown as overlay, where darker postures indicate a posture which is close in time to the via-point.

of

$$\lambda^* = \frac{\sum_{i \neq j} \widehat{\text{Var}}(R_{ij})}{\sum_{i \neq j} R_{ij}^2}. \tag{2.9}$$

The term $\widehat{\text{Var}}(R_{ij})$ denotes the variance of the elements of the matrix $\boldsymbol{R}$ which can be estimated from the samples and their weightings $\{\boldsymbol{\theta}^{[k]}, w^{[k]}\}_{k=1\ldots N}$ by

$$\widehat{\text{Var}}(R_{ij}) = \frac{\sum_{k=1}^N (w^{[k]})^2}{(1 - \sum_{k=1}^N (w^{[k]})^2)^3} \sum_{k=1}^N w^{[k]} (C_{ij}^{[k]} - \overline{C}_{ij})^2, \tag{2.10}$$

$$C_{ij}^{[k]} = \frac{(\theta_i^{[k]} - \mu_i)(\theta_j^{[k]} - \mu_j)}{\sqrt{S_{ii} S_{jj}}}, \quad \overline{C}_{ij} = \sum_{k=1}^N w^{[k]} C_{ij}^{[k]},$$

where $\mu_i = \sum_{k=1}^N w^{[k]} \theta_i^{[k]}$ is the mean of $i$th element of the parameter vector $\boldsymbol{\theta}$. For more details how to compute $\widehat{\text{Var}}(R_{ij})$ from samples, we refer to the appendix of [88].

### 2.3.2 Covariance Estimation with Controlled the Entropy Reduction

While the covariance shrinkage can already improve the performance of weighted ML algorithms, it still did not lead to fully satisfying results. Yet, in policy search, we can use more information as in standard density estimation. First, we know a good

prior upper level policy from which the unweighted samples have been generated from. Moreover, we know that the policy update should not reduce the entropy of the new upper level policy too quickly which leads to premature convergence. In our new algorithm, Covariance Estimation with Controlled Entropy Reduction (CECER), we combine the sample estimate of the covariance matrix $\boldsymbol{S}$ with the old covariance matrix $\boldsymbol{\Sigma}_q$ that has been used to generate the data, i.e.,

$$\boldsymbol{\Sigma} = \lambda \boldsymbol{\Sigma}_q + (1 - \lambda)\boldsymbol{S}.$$

The factor $\lambda \in [0, 1]$ is chosen in such a way that the entropy of the new upper level policy is reduced by a certain amount $\Delta H$. The entropy of a Gaussian distribution only depends on its covariance $\boldsymbol{\Sigma}$ and is given by

$$H(\boldsymbol{\Sigma}) = 0.5(p + p \log(2\pi) + |\boldsymbol{\Sigma}|).$$

Where $p$ is the dimension of the parameter space $\boldsymbol{\theta}$ and $|.|$ is the determinant operator. We choose $\lambda$ such that we achieve a desired entropy reduction, i.e.,

$$H(\boldsymbol{\Sigma}_q) - H(\lambda \boldsymbol{\Sigma}_q + (1 - \lambda)\boldsymbol{S}) = \Delta H.$$

We scale $\Delta H = \alpha \phi_{\text{eff}}$ with the number of effective samples $\phi_{\text{eff}}$ that have been used to compute the sample covariance $\boldsymbol{S}$, i.e., if more samples are available for the sample estimate, the entropy reduction can be higher. A higher entropy reduction leads to a smaller $\lambda$ value as we can rely more on our sample estimate. In order to find the correct $\lambda$ value, we applied a simple exhaustive search and we could always find the $\lambda$ with correct entropy reduction Algorithm. Algorithm 1 shows the Covariance Estimation with Controlled the Entropy Reduction (CECER) .

## 2.4 Experiments

We use the full covariance, the diagonal covariance and the covariance shrinkage algorithm and compare it to the CECER algorithm. The comparisons are done for the policy updates of REPS [55] and PI$^2$ [90][5]. Similar to Sep-CMA-ES [81], we call the algorithms with the diagonal matrix estimate, Sep-REPS and Sep-PI$^2$. We call the algorithms with shrinkage update, shrinkage-REPS and shrinkage-PI$^2$ respectively. CECER-REPS and CECER-PI$^2$ use CECER for policy update. We also compare

[5]The full covariance matrix update is the standard policy update method for episode version of REPS and PI$^2$

**Algorithm 1** Covariance Estimation with Controlled the Entropy Reduction

**Input :** Data Set $\mathcal{D}\{\theta^{[k]}, w^{[k]}\}_{k=1...N}$, the old covariance matrix $\boldsymbol{\Sigma}_q$ and the scaling factor $\alpha$ for entropy reduction

**Compute the sample covariance $\boldsymbol{S}$:**

$$\boldsymbol{\mu} = \sum_{i=1}^{N} w^{[k]} \boldsymbol{\theta}^{[k]}, \quad \boldsymbol{S} = \frac{\sum_{i=1}^{N} w^{[k]} (\boldsymbol{\theta}^{[k]} - \boldsymbol{\mu})^T (\boldsymbol{\theta}^{[k]} - \boldsymbol{\mu})}{1 - \sum_{i=1}^{N} \left(w^{[k]}\right)^2}.$$

**Compute the number of effective samples $\phi_{\text{eff}}$ and the entropy reduction $\Delta H$:**

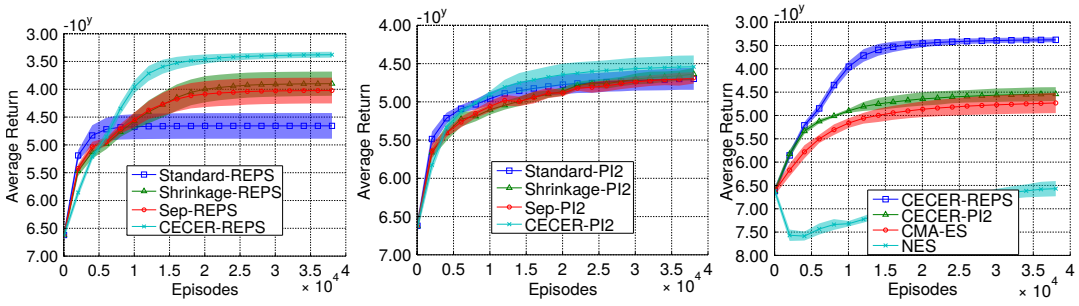
$$\phi_{\text{eff}} = \frac{1}{\sum_{k=1}^{N} (w^{[k]})^2}, \quad \Delta H = \alpha \phi_{\text{eff}}.$$

**Choose the $\lambda$ such that following equality is satisfied**

$$H(\boldsymbol{\Sigma}_q) - H(\lambda \boldsymbol{\Sigma}_q + (1 - \lambda)\boldsymbol{S}) = \Delta H.$$

**Compute the new covariance matrix $\boldsymbol{\Sigma}$:**

$$\boldsymbol{\Sigma} = \lambda \boldsymbol{\Sigma}_q + (1 - \lambda)\boldsymbol{S}.$$




(a) REPS (b) $PI^2$ (c) Stochastic Search Algorithms

FIGURE 2.3: The performance comparison for hole reaching task using a 5-link planar robot. The results show that CECER clearly outperforms the other covariance update methods for REPS and $PI^2$. Moreover CECER-REPS clearly outperforms the other stochastic search algorithms.

these algorithms to other state of the art methods in stochastic search such as CMA-ES [32] and NES [92]. For our comparisons, we used a multi-link planar robot with DMPs [41] as underlying lower level control policy. Each link had a length of 1m. We used 5 basis functions per degree of freedom for the DMPs. We use a 5-link planar robot that has to reach a given point in task space. We call this task *reaching task*. The resulting lower level policy has 25 parameters, but we also test the algorithms in high-dimensional parameter spaces by scaling up the robot to 20 links (100 parameters). This task has a relatively smooth reward function and is therefore

easy to learn. We make the task more difficult by introducing hard obstacles. We use the same planar robot to reach in a given hole on the ground, see Figure 2.2(c). Whenever the robot touches the ground with one of its links, a large penalty is added to the reward. Due to this discontinuity in the objective function, the task is much harder to learn. We call this task *hole reaching task*. For the hole reaching task, we used a 5-link and 15-link version of the robot, resulting in 30 parameters and 90 parameters lower-level policies to optimise. We compared the REPS and PI$^2$ algorithms with different policy updates individually and compared the best variant against CMA-ES and NES. In each iteration, we generated 40 new samples. For REPS and PI$^2$ we always keep the last $L = 400$ samples, while for NES and CMA-ES $40$ current samples are kept[6]. We show the average as well as the variance of the results over 10 trials for each experiment.

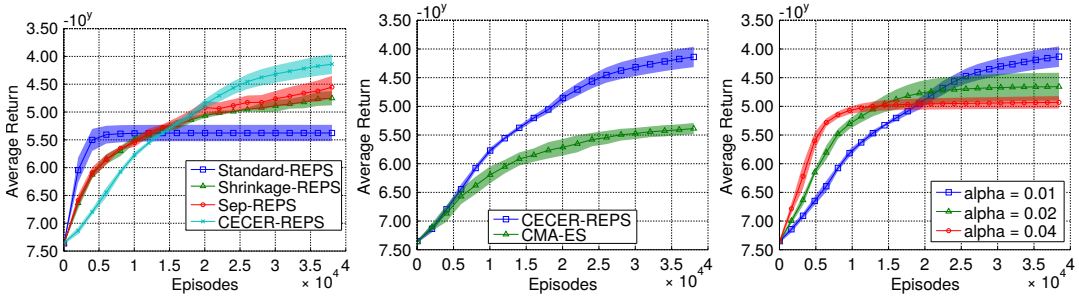
### 2.4.1 Planar Reaching Task

For completing the reaching task the robot has to reach a via-point $\boldsymbol{v}_{50} = [1, 1]$ at time step $50$ with its end-effector and at the final time step $T = 100$ the point $\boldsymbol{v}_{100} = [5, 0]$. The reward was given by a quadratic cost term for the two via-points as well as quadratic costs for high accelerations. The DMPs goal attractor for reaching the final state was assumed to be known. Hence, the parameter vector $\boldsymbol{\theta}$ for a 5-link robot with 5 basis function for each degree of freedom had $25$ dimensions. The results in Figure 2.1 show that CECER outperforms the other covariance estimation methods where CECER-REPS reach the average reward -1714 and shrinkage-REPS achieve an average reward of -2000. CECER-REPS has a better learning rate compare to the other methods. Yet, all the algorithms perform good in this task due to simplicity of the task. We also evaluated the same task with a 20-link planar robot, resulting in a 100 dimensional parameter space. The results in Figure 2.2 show that CECER and CECER-REPS clearly outperform the other methods.

### 2.4.2 Planar Hole Reaching Task

For completing the hole reaching task the robot end effector has to reach the bottom of a hole (35 cm wide and 1m deep) centred point $[2, 0]$ without any collision with the ground or the hole wall. The reward was given by a quadratic cost term for the desired final point, quadratic costs for high accelerations and quadratic costs for collisions with the environment. Note that this objective function is discontinuous due to the quadratic costs for collisions . The goal attractor for reaching the

[6]NES and CMA-ES algorithms typically only use the new samples and discard the old samples. We also tried keeping old samples which didn't lead to a better performance.




(a) REPS (b) Stochastic Search Algorithms (c) Evaluation of $\alpha$

FIGURE 2.4: The performance comparison for high dimensional hole reaching task using a 15-link planar robot. The results show that CECER clearly outperforms the other covariance update methods for REPS policy update and CECER-REPS outperforms the CMA-ES (c) It shows the performance of the CECER-REPS for three different entropy reduction scale factor $\alpha$. The bigger $\alpha$ results in more entropy reduction of the covariance matrix.

final state in this task is unknown and need to be learned. Hence, our lower level policy for a 5-link robot with 5 basis functions for each degree of freedom had 30 dimensions. The setup, including the learned policy is shown in Figure 2.2(c). The results in Figure 6.3 show that CECER has the best performance with significant difference. Covariance shrinkage performed the second best among all covariance estimation methods. We also see that CECER-REPS considerably outperforms the other methods. We also evaluated the same task with a 15-link planar robot, resulting in a 90 dimensional parameter space. The results in Figure 4 show that CECER and CECER-REPS clearly outperform the other methods with significant difference in performance. Using this task, we also evaluate the performance of CECER-REPS for three different $\alpha$ (Figure 4(c) ). It turns out with bigger $\alpha$ the search distribution shrinks faster, resulting in premature convergence. For large $\alpha$ values, the algorithm will only use the full sample covariance matrix, and thus, perform like the standard REPS algorithm with full covariance estimation.

## 2.5 Conclusion

In this chapter, we compared different methods for estimating the covariance matrix of a Gaussian policy for weighted ML based policy search methods. Weighted ML estimate of covariance matrices is an unreliable estimator with a high variance.

The use of WMLE leads to over-fitted covariance estimates, and, hence the variance/entropy of the policy decreases too quickly, which may cause premature convergence. We proposed a new algorithm called Covariance Estimation with Controlled the Entropy Reduction. We showed that using the CECER, we could control the entropy reduction of the policy and get a better covariance matrix approximation, which results in an significant improved performance of the policy search algorithm.

# Chapter 3

# Driving CMA-ES from Information Geometry

CMA-ES is one of the most popular stochastic search algorithms. It performs favourably in many tasks without the need of extensive parameter tuning. The algorithm has many beneficial properties, including automatic step-size adaptation, efficient covariance updates that incorporates the current samples as well as the evolution path and its invariance properties. Its update rules are composed of well established heuristics where the theoretical foundations of some of these rules are also well understood. However, a consistent mathematical framework for all CMA-ES update rules is missing so far. In this research we will fully derive all CMA-ES update rules within the framework of expectation-maximisation-based stochastic search algorithms using information-geometric trust regions. We show that the use of the trust region results in similar updates to CMA-ES for the *mean* and the *covariance* matrix while it allows for the derivation of an improved update rule for the step-size. Furthermore, the trust region also yields more stable updates for the *mean* that allow an application in higher dimensional parameter spaces. Our new algorithm, Trust-Region Covariance Matrix Adaptation Evolution Strategy(TR-CMA-ES) is fully derived from first order optimization principles and outperforms the standard CMA-ES algorithm in all tested scenarios.

## 3.1 Introduction

Stochastic search algorithms [32, 92] are black box optimizers of a fitness function that is either unknown or too complex to be modelled explicitly. These algorithms only use the fitness values and don't require gradients or higher derivatives of the fitness function. Stochastic search algorithms typically maintain a stochastic search distribution over individuals or candidate solutions, which is typically a Gaussian distribution. This search distribution is used to generate a population of individuals which are evaluated by their corresponding fitness values. Subsequently, a new

stochastic search distribution is computed by either computing gradient based updates [92], expectation-maximisation-based updates [54, 19], evolutionary strategies [32], the cross-entropy method [61] or information-theoretic policy updates [4], such that the individuals with higher fitness will have better selection probability.

The Covariance Matrix Adaptation - Evolutionary Strategy (CMA-ES) is one of the most popular stochastic search algorithms [32]. It performs well in many tasks and does not require extensive parameter tuning. There are three ingredients for the success of CMA-ES. Firstly, the covariance matrix update efficiently combines the old covariance matrix with the sample covariance. Secondly, the evolution path, which stores the average update direction is also incorporated in the covariance matrix to shape future update directions. Lastly, the step-size adaptation of CMA-ES scales the distribution efficiently, increasing it when subsequent updates are correlated while decreasing it otherwise. All these update rules are well established heuristics. The foundations of some of these heuristics are also already theoretical well understood. However, a unique mathematical framework that explains all these update rules is so far still missing.

In contrast, expectation maximisation-based algorithms[54, 61, 19] (Section 3.2.2) optimize a clearly defined objective, i.e., the maximization of a lower-bound. The maximisation of lower bound in each iteration is equivalent to weighted maximum likelihood estimation (MLE) of the distribution. It is well-known that ML estimation of the covariance matrix yields a degenerate, over-fitted distribution [5] which typically results in premature convergence of the algorithm. We extend the EM-based framework by incorporating information geometric trust regions. Intuitively, trust region will restrict the change of the search distribution in each update. We implement such trust region by bounding the Kullback-Leibler(KL) divergence[1] of the search distribution(Section 3.3). Using these trust regions, we can recover exactly the same form of covariance update as in CMA-ES.And the mean update of CMA-ES is a degenerate case of the trust region update. Furthermore, we can derive an improved step-size update rule that is based on the same theoretical foundation as the remaining update rules. Our new algorithm, Trust-Region Covariance Matrix Adaptation Evolution Strategy (TR-CMA-ES) outperforms the original algorithm in all our tested scenarios, which includes typical standard benchmark functions and complex policy optimization tasks from robotics.

[1] For simplicity we use 'KL' and 'KL divergence' interchangeably

### 3.1.1 Related Work

We will review CMA-ES and Expectation-Maximisation-based algorithms in the preliminaries section. In this section, we will briefly review other existing stochastic search algorithms.

The Natural Evolution Strategy (NES) uses the natural gradient to optimize the expected fitness value under the search distribution [92]. The natural gradient has been shown to outperform the standard gradient in many applications in machine learning [7]. The intuition of the natural gradient is that we want to obtain an update vector of the parameters of the search distribution that optimises the value of expected fitness while the KL-divergence between new and current search distributions is bounded. To obtain this update vector, a second order approximation of the KL, which is equivalent to the Fisher information matrix, is used. The resulting natural gradient is obtained by multiplying the standard gradient with the inverse of the Fisher matrix. Yet, in contrast to our algorithm, NES family algorithms do not exactly enforce a desired bound of the KL-divergence but use a constant learning rate [92].

The use of trust regions is a common approach in policy search to obtain a stable policy update and to avoid premature convergence [75, 4, 86]. The model-based relative entropy stochastic search algorithm (MORE)[4] uses a local quadratic surrogate to update it's Gaussian search distribution. As these surrogates can be inaccurate, MORE doesn't exploit the surrogate greedily, but uses a trust region in form of a KL-bound. This trust region defines how much the algorithm can exploit the quadratic model. Moreover, MORE explicitly bounds the entropy loss to avoid premature convergence. This method has been shown very competitive when the hyper parameters are set correctly. Similar trust regions have been used in other policy search methods such as Trust Region Policy Optimization [86] and Relative Entropy Policy Search [75].

Similar to these methods, our framework also uses a KL bound to define a trust region. However, in difference to all these policy search methods, firstly we use this bound in an EM-based framework and secondly we use the reverse KL bound. As the KL is not symmetric, this is a very important difference. For a more detailed discussion please see Section 3.3.

## 3.2 Preliminaries

In stochastic search, we want to maximize a fitness function $f(\boldsymbol{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$. The goal is to find one or more individuals $\boldsymbol{x} \in \mathbb{R}^n$ which have the highest possible fitness

value. The only accessible information is the fitness values of the individuals. Typically stochastic search algorithms, maintain a Gaussian distribution on individuals,

$$\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}, \sigma\boldsymbol{\Sigma}\right),$$

where $\boldsymbol{\mu} \in \mathbb{R}^n$, $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ and $\sigma \in \mathbb{R}^+$ respectively are the mean(our current guess of optimum), the covariance matrix and step size which together define the exploration of the search distribution. We seek to find a distribution over individuals $\boldsymbol{x}$, denoted $\pi(\boldsymbol{x};\boldsymbol{\theta})$, that minimises the expected fitness

$$\mathbb{E}[f(\boldsymbol{x})] = \int_{\boldsymbol{x}} \pi(\boldsymbol{x};\boldsymbol{\theta}) f(\boldsymbol{x}) d\boldsymbol{x}. \tag{3.1}$$

At each new iteration $t+1$, $N$ individuals $\boldsymbol{x}$ are drawn from the current Gaussian distribution $\pi(\boldsymbol{x}, \boldsymbol{\theta}_t)$, with $\boldsymbol{\theta}_t = \{\boldsymbol{\mu}_t, \sigma_t, \boldsymbol{\Sigma}_t\}$. These individuals are evaluated and, with their fitness values, construct a dataset $\{\boldsymbol{x}_i, f(\boldsymbol{x}_i)\}_{i=1\dot{N}}$ that is subsequently used to compute a new Gaussian search distribution $\pi_{\boldsymbol{\theta}_{t+1}}$, with

$$\boldsymbol{\theta} = \{\boldsymbol{\mu}_{t+1}, \sigma_{t+1}, \boldsymbol{\Sigma}_{t+1}\}$$

such that better individuals will have higher selection probability.

### 3.2.1 Covariance Matrix Adaptation - Evolutionary Strategy

As CMA-ES will be our main baseline for comparisons, in this section we briefly explain the update rules of the CMA-ES algorithm to obtain a new search distribution.

**Fitness Transformation.** CMA-ES uses a rank preserving transformation of fitness values which makes it invariant under monotone transformations of the fitness function[31]. In particular, CMA-ES gives zero weights to the worse half of the population and the weight of the $jth$ best individuals is set to

$$w_j = \ln(N/2 + 0.5) - \ln(j). \tag{3.2}$$

We will use the same weighting method in our algorithm.

**Mean Update Rule.** Using the weight transformation, the next distribution $\boldsymbol{\mu}_{t+1}$ is set to the weighted sum of the individuals, i.e,

$$\boldsymbol{\mu}_{t+1} = \sum_{i=1}^{N/2} w_i \boldsymbol{x}, \quad \sum_{i=1}^{N/2} w_i = 1.$$

**Evolution Path.** The evolution path records the sum of consecutive update steps, i.e, $\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_t$. If consecutive update steps are towards the same direction, i.e., they are correlated, their updates will sum up. In contrast, if they are decorrelated, the update directions cancel each other out. Using the information from the evolution path leads to significant improvements in terms of convergence speed, as it enables the algorithm to exploit correlations between consecutive steps.

**Covariance Matrix Update Rule.** The covariance matrix update is computed using the local information about the fitness function from the individuals, called the in rank-$\mu$ update, and the information about the evolution of the distribution contained in the evolution path $\boldsymbol{p}_{ct+1}$, called the rank-1 update. The covariance update is defined as

$$\boldsymbol{\Sigma}_{t+1} = (1 - c_{\boldsymbol{\mu}} - c_1)\boldsymbol{\Sigma}_t + \underbrace{c_{\mu} \sum_{i=1}^{N} \frac{w_i(\boldsymbol{x}_i - \boldsymbol{\mu}_t)(\boldsymbol{x}_i - \boldsymbol{\mu}_t)^T}{\sigma_t}}_{\text{Rank-}\boldsymbol{\mu}\text{ update}} + \underbrace{c_1 \boldsymbol{p}_{ct+1} \boldsymbol{p}_{ct+1}{}^T}_{\text{Rank-one update}},$$

where $c_1 \geq 0$ and $c_{\boldsymbol{\mu}} \geq 0$ are learning rates which are set by well established heuristics.

**Step Size Update Rule.** Subsequently, the evolution path is also used to find a new step size $\sigma_{t+1}$ such that the difference between the distributions of the actual evolution path and an evolution path under random selection is minimised, see [31] for more details.

**Theoretical Foundation.** Recently there are theoretical studies to justify the update rules of CMA-ES. FOr example, In [74] and [6] it has been shown that the CMA-ES mean and rank-$\mu$ update rules can be derived by computing the natural gradient of the expected fitness under the search distribution. Moreover, Akimoto et al. [6] nicely sketche the relation between CMA-ES and the EM-based stochastic search framework conceptually. However, it does not define a concrete mathematical framework for driving the search distribution update rules. While these theoretical studies provide a great understanding of CMA-ES, to the best of our knowledge, they do not lead to an improvement over the original CMA-ES. The reason is that, these frameworks [74], [6] do not derive the rank-1 of CMA-ES and a step-size update rule, which are crucial features of CMA-ES. Our new algorithm will have all the features of CMA-ES while it will be fully derived from a well defined principle.

### 3.2.2 Stochastic Search by Expectation-Maximisation

An alternative view to the formulation in Equation 3.1 is to convert the problem into an inference problem [72]. In order to use inference, the common method is to define a binary fitness event $R$ as observed variable. To simplify notation we will always write $R$ when we mean $R = 1$. The probability of this fitness event is given by $p(R|\boldsymbol{x}) \propto C(f(\boldsymbol{x}))$, where $C$ is a monotonically increasing (i.e., rank preserving) transformation of the fitness function $f(\boldsymbol{x})$. Intuitively $p(R|\boldsymbol{x})$ defines the probability that individual $\boldsymbol{x}$ is the optimum solution. Hence now we optimise the objective

$$p(R;\boldsymbol{\theta}) = \int_{\boldsymbol{x}} p(R|\boldsymbol{x})\pi(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x}, \tag{3.3}$$

where $p(R;\boldsymbol{\theta})$ is the marginal distribution of the event $R$. We would now like to find the distribution parameters $\boldsymbol{\theta}$, that maximises $p(R;\boldsymbol{\theta})$. We can also maximize any strictly increasing function of $p(R;\boldsymbol{\theta})$. In particular, the derivations will be simpler if instead we maximise the log probability of the event $R$, i.e,

$$\log p(R;\boldsymbol{\theta}) = \log \int_{\boldsymbol{x}} p(R|\boldsymbol{x})\pi(\boldsymbol{x};\boldsymbol{\theta})d\boldsymbol{x} \tag{3.4}$$

To optimise $\log p(R;\boldsymbol{\theta})$, we resort to an iterative expectation maximization (EM) algorithm [19, 72]. In each iteration, the EM algorithm iteratively constructs a lower bound on $\log p(R;\boldsymbol{\theta})$ (E-step) and then optimizes that lower bound to obtain a new search distribution $\pi_{t+1} = \pi(\boldsymbol{x};\boldsymbol{\theta}_{t+1})$ (M-step).

**Constructing the lower bound (E-Step).** Similar to prior work [72], we can now introduce a variational distribution $q(\boldsymbol{x})$ which is used to decompose the $\log p(R;\boldsymbol{\theta})$, i.e,

$$\log p(R;\boldsymbol{\theta}) = (q,\boldsymbol{\theta}) + \mathrm{KL}(q(\boldsymbol{x})||p(\boldsymbol{x}|R;\boldsymbol{\theta})), \tag{3.5}$$

where the first term

$$(q,\boldsymbol{\theta}) = \int_{x} q(\boldsymbol{x}) \log \frac{p(R|\boldsymbol{x})\pi(\boldsymbol{x};\boldsymbol{\theta})}{q(\boldsymbol{x})} d\boldsymbol{x} \tag{3.6}$$

is the lower bound of $\log p(R;\boldsymbol{\theta})$ and second term

$$\mathrm{KL}(q(\boldsymbol{x})||p(\boldsymbol{x}|R;\boldsymbol{\theta})) = -\int_{\boldsymbol{x}} q(\boldsymbol{x}) \log \frac{p(\boldsymbol{x}|R;\boldsymbol{\theta})}{q(\boldsymbol{x})} d\boldsymbol{x} \tag{3.7}$$

is the KL-divergence between the variational distribution $q(\boldsymbol{x})$ and the posterior

$$p(\boldsymbol{x}|R;\boldsymbol{\theta}) = \frac{p(R|\boldsymbol{x})\pi(\boldsymbol{x};\boldsymbol{\theta})}{\int p(R|\boldsymbol{x})\pi(\boldsymbol{x};\boldsymbol{\theta})}, \tag{3.8}$$

which is proportional to the search distribution $\pi(\boldsymbol{x};\boldsymbol{\theta})$ weighted by fitness event probabilities $p(R|\boldsymbol{x})$. Equation 3.5 holds for any variational distribution $q(\boldsymbol{x})$ and parameters $\boldsymbol{\theta}$, and hence,

$$\log p(R;\boldsymbol{\theta}) \geq (q(\boldsymbol{x}),\boldsymbol{\theta}) \tag{3.9}$$

as the KL term is always positive. Note that the lower bound will be tight at our current distribution parameters $\boldsymbol{\theta}_t$, i.e., $\log p(R;\boldsymbol{\theta}_t) = (q(\boldsymbol{x}),\boldsymbol{\theta}_t)$, if we choose a variational distribution $q(\boldsymbol{x})$ such that $\mathrm{KL}(q(\boldsymbol{x})||p(\boldsymbol{x}|R;\boldsymbol{\theta}_t)) = 0$, which is equivalent to choosing

$$q(\boldsymbol{x}) = p(\boldsymbol{x}|R;\boldsymbol{\theta}_t) = \frac{p(R|\boldsymbol{x})\pi(\boldsymbol{x};\boldsymbol{\theta}_t)}{\int_{\boldsymbol{x}} p(R|\boldsymbol{x})\pi(\boldsymbol{x};\boldsymbol{\theta}_t)d\boldsymbol{x}}.$$

This choice of the $q(\boldsymbol{x})$ gives us a lower-bound $(p(\boldsymbol{x}|R;\boldsymbol{\theta}_t),\boldsymbol{\theta})$ on the $\log p(R;\boldsymbol{\theta})$ such that $p(R;\boldsymbol{\theta}_t) = (p(\boldsymbol{x}|R;\boldsymbol{\theta}_t),\boldsymbol{\theta}_t)$. Constructing this lower-bound corresponds to the E-step.

**Optimising the lower bound (M-Step).** In the M-step, we then maximise $(p(\boldsymbol{x}|R;\boldsymbol{\theta}_t),\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$ to obtain a new parameters $\boldsymbol{\theta}_{t+1}$. Typically, $p(R|\boldsymbol{x})$ is unknown and we only have access to sample evaluations for the individual $\boldsymbol{x}$ generated from the search distribution $\pi(\boldsymbol{x};\boldsymbol{\theta}_t)$. In this case, the lower-bound can be approximated by

$$\begin{aligned}(q(\boldsymbol{x}),\boldsymbol{\theta}) &\approx 1/N \sum_i \frac{q(\boldsymbol{x}_i)}{\pi(\boldsymbol{x}_i;\boldsymbol{\theta}_t)} \log \frac{p(R|\boldsymbol{x}_i)\pi(\boldsymbol{x}_i;\boldsymbol{\theta})}{q(\boldsymbol{x}_i)} d\boldsymbol{x} \\ &\approx 1/N \sum_i w_i \log \pi(\boldsymbol{x}_i;\boldsymbol{\theta}) + \text{const.},\end{aligned} \tag{3.10}$$

where $w_i \propto p(R|\boldsymbol{x}_i)$. This corresponds to a weighted maximum likelihood estimate.

**Monotonic improvement guarantee.** Expectation-Maximization stochastic search methods inherit a monotonic improvement guarantee from the EM algorithm, i.e., $\log p(R;\boldsymbol{\theta}_t)$ is always increased (or stays the same) at each iteration. This is easy to see by noting that the lower-bound $\mathcal{L}$ is tight after the E-step, i.e., $\log p(R;\boldsymbol{\theta}_t) = (q,\boldsymbol{\theta}_t)$. Therefore, optimising the lower-bound at the M-step, can only improve $\log p(R;\boldsymbol{\theta}_{t+1})$ over $\log p(R;\boldsymbol{\theta}_t)$.

**Disadvantage of EM-based algorithms.** While these algorithm can improve the

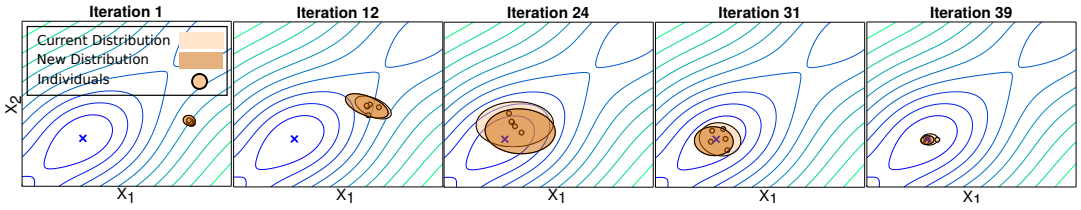



FIGURE 3.1: In this illustration, we use a slightly modified version of the McCormick function with two dimensions. The blue cross shows the optimum and blue contours have higher value than green contours. In iteration one, the algorithm starts with a small distribution. In each iteration we sample five individuals. The figures show that our step size strategy effectively controls growing and shrinking of the distribution. At the beginning, it naturally grows and when it has found the optimum, it naturally shrinks (iteration 24-39). Please note that the step size strategy is derived from a well defined consistent principle.

search distribution, it can quickly result in a degenerated distribution which stops exploration and would lead to premature convergence, which is a problem in many of these methods [61, 54]. The cause of this limitation is mainly the maximum likelihood estimate (Equation 3.10) which over-fits the current individuals and change the current search distribution drastically[5].

## 3.3 Trust Regions for Covariance Matrix Adaptation

In order to remedy the problem of premature convergence in EM-based algorithms, we will add information-geometric trust region to the optimization objective of the lower bound. The trust region regularizes the maximization step and bound the entropy loss of the new distribution with respect to the current distribution. As we do not fully maximize the lower-bound any more, our algorithm belongs to the class of generalized Expectation-Maximisation algorithms [22] which also holds the expectation improvement guarantee of conventional EM algorithms .

Our constraint optimization problem for optimizing the lower bound now is given by

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_i w_i \log \pi(\boldsymbol{x}_i; \boldsymbol{\theta}) \\ &\text{s.t. } \mathrm{KL}\big(\pi(\boldsymbol{x}; \boldsymbol{\theta}_t) || \pi(\boldsymbol{x}; \boldsymbol{\theta})\big) \leq \epsilon, \end{aligned} \tag{3.11}$$

where $\epsilon$ defines the bound on the KL divergence. With this bound we can define a trust region for our updates, Intuitively by holding the search distribution in this trust

region we can avoid a degenerated distribution. In next section, we will show how to solve this constraint optimisation problem efficiently in closed form, to obtain the update rules for the new Gaussian search distribution. The resulting update strategies match the update strategies of CMA-ES except for small, but important differences as we will discuss later in this section. While bounding the KL-divergence is a well established method [75, 59, 4, 86], it has so far not been applied to expectation-maximisation stochastic search algorithms. Another interesting observation is that, in difference to all existing policy search methods that use a trust region [75, 4], we employ the reverse KL divergence measure, i.e., instead of bounding $\mathrm{KL}(\pi||\pi_t)$ we bound $\mathrm{KL}\big(\pi_t||\pi\big)$. Hence, our derivation reveals an interesting connection of CMA-ES to many trust region optimization algorithms that use the forward KL, $\mathrm{KL}(\pi||\pi_t)$ which was so far unknown. Note that in the case of a Gaussian search distribution, we can solve the optimisation program in equation 3.11 in closed form only if we use the reverse KL. Moreover, there is a tight connection to Bayesian estimation where the current distribution $\pi_t$ can be used as prior. With the difference that in our framework, the influence of the prior is controlled by the trust region parameter $\epsilon$.

### 3.3.1 Update Rules for Multivariate Normal Distributions

In each iteration, we solve the optimization program given in Equation 3.11 with

$$\pi(\boldsymbol{x};\boldsymbol{\theta}) = (2\pi)^{-n/2}|\boldsymbol{\Sigma}|^{-0.5}\exp(-0.5(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))$$

and the KL divergence between two Gaussian distributions

$$\begin{aligned}\mathrm{KL}\big(\pi_t||\pi\big) = 0.5\Bigg(&\mathrm{tr}(\sigma^{-1}\boldsymbol{\Sigma}^{-1}\sigma_t\boldsymbol{\Sigma}_t) \\ &+ (\boldsymbol{\mu}-\boldsymbol{\mu}_t)^T\sigma^{-1}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_t) - k + \ln\frac{|\sigma\boldsymbol{\Sigma}|}{|\sigma_t\boldsymbol{\Sigma}_t|}\Bigg).\end{aligned} \tag{3.12}$$

Where $|.|$ and $Tr(.)$ are determinant and trace operators respectively. In general, to solve this constraint optimisation problem, we first construct the Lagrangian equation, i.e,

$$L(\eta,\boldsymbol{\theta}) = \sum_i w_i \log\pi(\boldsymbol{x}_i;\boldsymbol{\theta}) + \eta(\epsilon - \mathrm{KL}\big(\pi_t||\pi\big)),$$

where $\eta$ is a Lagrangian multiplier.

Subsequently, we differentiate $L(\eta,\boldsymbol{\theta})$ we respect to $\boldsymbol{\theta}$ and set it to zero to obtain $\boldsymbol{\theta}^*$ which will depend on the Lagrangian multiplier $\eta$. To compute the optimal value

for the Lagrangian multiplier $\eta^*$, we can optimize the dual function $L(\eta, \boldsymbol{\theta}^*)$ which is obtained by setting the optimal $\boldsymbol{\theta}^*$ back into the Lagrangian.

In order to perform an efficient optimization and to obtain the CMA-ES updates, we employ a coordinate descent strategy where we optimize for each parameter, i.e, the new mean $\boldsymbol{\mu}_{t+1}$, the covariance matrix $\boldsymbol{\Sigma}_{t+1}$ and the step size $\sigma_{t+1}$, independently. I.e., when we optimize, for example, for the covariance matrix $\boldsymbol{\Sigma}$, we use the current mean $\boldsymbol{\mu}_t$ and step size $\sigma_t$ for the remaining parameters. This decoupling have three advantages. Firstly, it renders the problem to a concave optimisation problem for each search distribution component. This is easy to verify by observing that the second derivative of the Lagrangian $L(\eta, \boldsymbol{\theta})$ with respect to each parameter is negative definite. Moreover, this decoupling allows us to set different $\epsilon$ values for each component, i.e., $\epsilon_{\boldsymbol{\mu}}, \epsilon_{\boldsymbol{\Sigma}}, \epsilon_{\sigma}$ for the mean, the covariance matrix and the step size respectively. Different $\epsilon$ lead to different learning rates. We can set a higher learning rate for the mean and step size than for the covariance as the covariance matrix has many more parameters which makes it more frail to overfit. Finally, by setting the mean to the current mean when optimising for covariance matrix and step size, we find a covariance matrix $\boldsymbol{\Sigma}_{t+1}$ and step size $\sigma_{t+1}$ that increases the likelihood of successful steps instead of likelihood of successful individuals. CMA-ES also uses the current mean when obtaining the new covariance matrix. This approach has been shown to be less prone to premature convergence [31].

### Mean Update Rule

Now, we construct the Lagrangian for obtaining the new mean, i.e,

$$L(\boldsymbol{\mu}, \eta) = 2\epsilon_{\mu}\eta - \text{tr}(\boldsymbol{\Sigma}_t{}^{-1}\boldsymbol{F}), \tag{3.13}$$

where

$$\boldsymbol{F} = \sum_i w_i(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T + \eta(\boldsymbol{\mu} - \boldsymbol{\mu}_t)(\boldsymbol{\mu} - \boldsymbol{\mu}_t)^T.$$

To obtain the new mean $\boldsymbol{\mu}_{t+1} = \text{argmax}_{\boldsymbol{\mu}}\, L(\boldsymbol{\mu}, \eta)$, we differentiate the Lagrangian with respect to $\boldsymbol{\mu}$ and set it to zero to obtain

$$\boldsymbol{\mu}_{t+1} = \frac{\eta^*_{\mu}\boldsymbol{\mu}_t + \sum w_i \boldsymbol{x}_i}{\sum w_i + \eta^*_{\mu}}, \tag{3.14}$$

where $\eta^*_{\mu}$ is the optimum Lagrangian multiplier which can be obtained by minimising the convex dual function

$$g_{\boldsymbol{\mu}}(\eta) = \text{supp}_{\boldsymbol{\mu}} L(\eta, \boldsymbol{\mu}) = L(\eta, \boldsymbol{\mu}_{t+1}),$$

i.e., $\eta_\mu^* = \text{argmin}_\eta\, g_{\boldsymbol{\mu}}(\eta)$. As indicated in the equation, the dual function is obtained by setting the optimal solution for the mean back into the Lagrangian. The dual function is convex and can be efficiently optimised by any arbitrary non-linear optimiser. We use fmincon tool in matlab (please see the matlab code for implementation).

If we set a large KL bound for the mean, i.e., $\epsilon_\mu \to \infty$, $\eta$ will go to 0 and we obtain the original CMA-ES update $\boldsymbol{\mu}_{t+1} = \sum w_i \boldsymbol{x}_i / \sum w_i$, which is equivalent to the unregularized weighted maximum likelihood estimate. While not regularizing the mean update works well for a moderate dimensionality of the parameter space using a rather high number of samples, it significantly degrades the performance for high-dimensional parameter spaces or small population sizes. As we will show, the regularized solution still works well in these cases.

### Incorporating the Evolution Path

The evolution path is a crucial ingredient of the performance of CMA-ES [31]. It summarizes the path taken by the recent distribution updates and can be interpreted as sort of momentum term. In order to exploit the evolution path in our formulation, we treat the evolution path $\boldsymbol{p}_c$ as an individual sample, i.e., our optimisation problem is now given by

$$\begin{aligned} &\max_{\boldsymbol{\theta}} \sum_i w_i \log \pi(\boldsymbol{x}_i;\boldsymbol{\theta}) + \lambda \log \pi(\boldsymbol{\mu}_t + \boldsymbol{p}_c;\boldsymbol{\theta}) \\ &\text{s.t. } \text{KL}\big(\pi(\boldsymbol{x};\boldsymbol{\theta}_t)||\pi(\boldsymbol{x};\boldsymbol{\theta})\big) \leq \epsilon, \end{aligned} \tag{3.15}$$

where $\lambda > 0$ is a user-defined weight that specifies the importance of the evolution path $\boldsymbol{p}_c$ with respect to other individuals. Empirically, we found it is better to use different $\lambda$ coefficients for the covariance matrix and the step size, i.e, $\lambda_{\boldsymbol{\Sigma}}$, $\lambda_\sigma$. Please note that we also tried to use the evolution path to adapt the mean as well. However, we observed that the learning process gets considerably unstable due to overshooting the optimum.

### Covariance Matrix Update Rule

We construct the covariance matrix Lagrangian for the optimisation program in Equation 3.15, i.e.,

$$\begin{aligned} L(\eta,\boldsymbol{\Sigma}) = &-(1+\lambda_{\boldsymbol{\Sigma}}+\eta)\ln|\boldsymbol{\Sigma}| + \eta(2\epsilon_{\boldsymbol{\Sigma}} + n + \ln|\boldsymbol{\Sigma}_t|) \\ &- \text{tr}(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Sigma}_s/\sigma_t + \eta\boldsymbol{\Sigma}_t + \lambda_{\boldsymbol{\Sigma}}\boldsymbol{p}_c\boldsymbol{p}_c^{\,T}/\sigma_t)), \end{aligned} \tag{3.16}$$

where

$$\boldsymbol{\Sigma}_s = \sum_i w_i(\boldsymbol{x}_i - \boldsymbol{\mu}_t)(\boldsymbol{x}_i - \boldsymbol{\mu}_t)^T. \tag{3.17}$$

To obtain the new covariance matrix $\boldsymbol{\Sigma}_{t+1} = \text{argmax}_{\boldsymbol{\Sigma}} L(\boldsymbol{\Sigma}, \eta)$, we differentiate the Lagrangian with respect to $\boldsymbol{\Sigma}^{-1}$ and set it to zero to obtain

$$\boldsymbol{\Sigma}_{t+1} = \frac{\eta^*_{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}_t + \boldsymbol{\Sigma}_s/\sigma_t + \lambda_{\boldsymbol{\Sigma}}\boldsymbol{p}_c\boldsymbol{p}_c^T/\sigma_t}{\sum_i w_i + \lambda_{\boldsymbol{\Sigma}} + \eta^*_{\boldsymbol{\Sigma}}} \tag{3.18}$$

in closed form. The coefficient $\eta^*_{\boldsymbol{\Sigma}}$ is again the optimum Lagrangian multiplier. It can be obtained by minimising the dual function $\eta^*_{\boldsymbol{\Sigma}} = \text{argmin}_\eta\, g_{\boldsymbol{\Sigma}}(\eta)$.

Now, by changing variables, i.e.,

$$\begin{aligned} 1 - \alpha - \gamma &= \frac{\eta}{\sum_i w_i + \lambda_{\boldsymbol{\Sigma}} + \eta}\ ,\ \gamma = \frac{\lambda_{\boldsymbol{\Sigma}}}{\sum_i w_i + \lambda_{\boldsymbol{\Sigma}} + \eta} \\ \alpha &= \frac{1}{\sum_i w_i + \lambda_{\boldsymbol{\Sigma}} + \eta}, \end{aligned} \tag{3.19}$$

we can rewrite the equation for $\boldsymbol{\Sigma}$ as

$$\boldsymbol{\Sigma}_{t+1} = (1 - \alpha - \gamma)\boldsymbol{\Sigma}_t + \alpha\frac{\boldsymbol{\Sigma}_s}{\sigma_t} + \gamma\frac{\boldsymbol{p}_c\boldsymbol{p}_c{}^T}{\sigma_t}.$$

Note that as $\eta > 0$ and $\lambda_{\boldsymbol{\Sigma}} > 0$ we can infer that $0 \leq \alpha + \gamma \leq 1$. Hence, we obtained the exact form of covariance matrix update of CMA-ES where the second and third terms are called rank $-\ \mu$ and rank-1 updates respectively. The only difference is that $\alpha$ and $\gamma$ are constant coefficients in CMA-ES. Please note that CMA-ES uses constant coefficients to combine the old covariance matrix with rank-1 and rank-$\mu$ while TR-CMA-ES compute these coefficients based on satisfying an exact KL bound. We observed that this exact KL-bound results in different coefficients in each iteration.

### Step Size Update Rule

We also construct the step size Lagrangian for the optimisation program in Equation 3.15, i.e.,

$$\begin{aligned} L(\eta, \sigma) = &-(1 + \lambda_\sigma + \eta)\ln\sigma^n + \eta(2\epsilon_\sigma + n + \ln({\sigma_t}^n)) \\ &- \sigma^{-1}\,\text{tr}({\boldsymbol{\Sigma}_t}^{-1}(\lambda_\sigma\boldsymbol{p}_c\boldsymbol{p}_c{}^T + \boldsymbol{\Sigma}_s) + \eta\sigma_t I). \end{aligned} \tag{3.20}$$

To obtain the new step size $\sigma_{t+1} = \text{argmax}_\sigma \, L(\sigma, \eta)$ we differentiate the Lagrangian with respect to $\sigma^{-1}$ and set it to zero to obtain

$$\sigma_{t+1} = \frac{n\eta_\sigma^* \sigma_t + \text{tr}(\boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\Sigma}_s + \lambda_\sigma \boldsymbol{p}_c \boldsymbol{p}_c^T))}{n(\sum w_i + \lambda_\sigma + \eta_\sigma^*)}. \tag{3.21}$$

Where $\boldsymbol{\Sigma}_s$ is given in equation 3.17 and the coefficient $\eta_\sigma^*$ is the optimal Lagrangian multiplier. It is again obtained by minimising the dual function $\eta_\sigma^* = \text{argmin}_\eta \, g_\sigma(\eta)$. In contrast to the current step size control approaches, our algorithm naturally increases or decreases the step size $\sigma$ based on the the current individuals and the evolution path without explicitly defining any heuristic. More formally, the step size is adapted to increase the likelihood of successful steps and the evolution path using the same mathematical foundations as the updates for the mean and evolution path. Figure 3.1 illustrates the effectiveness our search distribution step size update rule. Now, all the updates follow the same principle.

### 3.3.2 Algorithm

Algorithm 1 shows a compact representation of the new stochastic search method. In each iteration, we generate N individuals $\boldsymbol{x} \in \mathbb{R}^n$ and evaluate their fitness values (Lines 1-7). Subsequently, we compute a weight for each individual based on the fitness value (Line 8). For ease of comparison we use the same rank preserving fitness transformation as CMA-ES. Please see preliminary section for the weighting method. We compute the number of effective samples (Line 9). Similar to CMA-ES, we empirically obtained a default setting for hyper parameters of the algorithm which scales with the dimension of the problem and size of the population (Line 10). Therefore user only needs to define size of the population N. Subsequently we compute the new mean and new evolution path (Line 11-15). Finally, we obtain the new covariance matrix and new step size (Line 16-21). Note that the $\eta^*$ are obtained by optimising the dual functions in previous section.

## 3.4 Experiments

In this section, we present our experimental evaluations of TR-CMA-ES in comparison to CMA-ES. We use two sets of benchmarks, the standard functions and three simulated robotics tasks where we compare against CMA-ES and also evaluate the different components of our new algorithm. For all experiments we use the same hyper parameter setting as given in Algorithm 1. For comparison, we use the CMA-ES

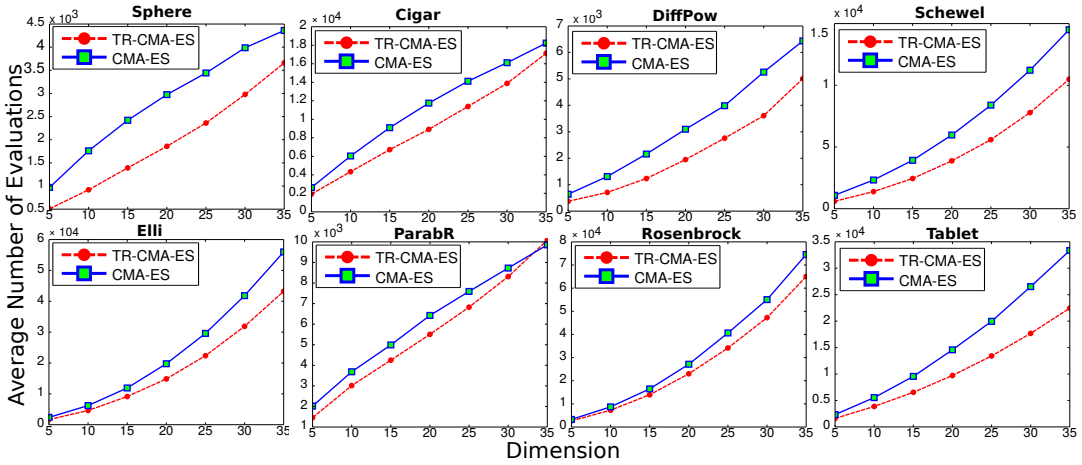



FIGURE 3.2: Plot of the average number of required fitness evaluations over 20 trials to reach the target fitness value of $-10^{-5}$ ($10^3$ for the unbounded function ParabR) for 8 different benchmark functions with 5 to 35 dimensions. TR-CMA-ES outperformed CMA-ES consistently for all benchmark functions. CMA-ES prematurely converged four times over all Rosenbrock experiments which we removed from the averaging computation. TR-CMA-ES didn't suffer from any premature convergence.

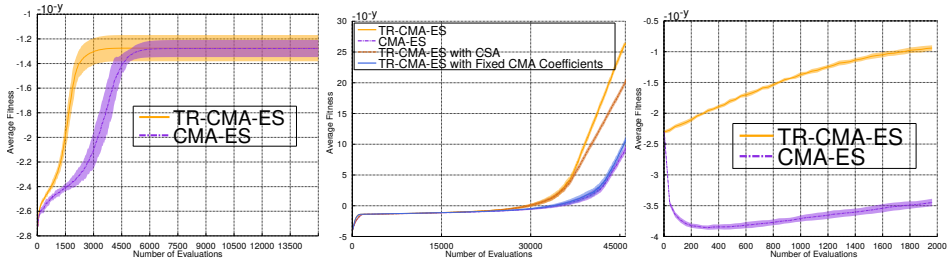
Matlab source code released in CMA-ES official website [2]. We run both algorithms for several trials for each single experiment. For both algorithms, we used the same population size and both algorithms will start with a same initial distribution in each trial. However, initial distribution for each trial is varied slightly. Also, both algorithms use fixed hyper parameter settings, i.e, throughout the experiments for both algorithms, we don't set any hyper parameters by hand[3].

### 3.4.1 Standard Functions

We empirically evaluate TR-CMA-ES on 9 benchmark functions from [32]. As TR-CMA-ES is a maximiser we multiply all the functions values by $-1$ to turn their minimum to a maximum. Now, except for PrabR function that is unbounded, all other functions have a global maximum of zero. In each iteration, we sample 30 individuals. For each trial, we randomly generate the mean of the initial distribution from a normal distribution with zero mean. We set the initial covariance matrix as a identity matrix with initial $\sigma = 1$. We ran TR-CMA-ES and CMA-ES on these functions with dimensions from 5 to 35 and a target fitness of $-10^{-5}$ ($10^3$ for ParabR). We perform 20 trials for each dimension and report the average number of fitness evaluations to reach the target fitness.

[2] https://www.lri.fr/~hansen/cmaes_inmatlab.html

[3] **Matlab source code of TR-CMA-ES (with examples) as well as videos regarding the robotics experiments are available at** https://goo.gl/9u15Wf




(a) Rastragin (b) Rosenbrock (c) Sphere(High Dimensional)

FIGURE 3.3: Comparison of CMA-ES and TR-CMA-ES on (a)Rastragin with 25 dimensions and TR-CMA-ES with step size from CMA-ES (TR-CMA-ES with CSA) and TR-CMA-ES with constant coefficients. on (b) Rosenbrock function with 25 dimensions. TR-CMA-ES always outperforms CMA-ES. On the Rosenbrock function, we also evaluated the significance of the new step-size update rule (with CSA) and TR-CMA-ES with constant coefficients instead of optimized coefficients from the trust region. While both variants still outperform the standard CMA-ES algorithm, the results show that both components are important ingredients for the superior performance of our algorithm. (c) We also compare our algorithm with CMA-ES on optimising a 200 dimensional sphere function where we sample only 2 individuals at each iteration. CMA-ES diverges as it does not use a regularization for the mean update, and can only slightly recover after the variance has shrunken for a large amount. TR-CMA-ES still converges. Please note that y in $-10^{-y}$ is value on y axis.

**Results.** Figure 3.2 provides the full results on the set of eight benchmark functions. The results show that TR-CMA-ES constantly outperforms CMA-ES in terms of number of evaluations to reach the target fitness. Please note that on the Rosenbrock function, CMA-ES prematurely converged four times while TR-CMA-ES didn't have any premature convergence issue. We removed those trials as outliers from our averaging computations. We also show the learning curve of optimising the (multimodal) Rastragin function in Figure 3.3(a) with the same setup as the other functions. Here, TR-CMA-ES is faster and also in average finds slightly better solution.

**Evaluating the TR-CMA-ES Components.** We also used a 25 dimensional Rosenbrock function to evaluate the step-size strategy and exact KL trust region of TR-CMA-ES. We use the same experimental setup as before and tested two more hybrid algorithms to compare with CMA-ES and TR-CMA-ES. For the first algorithm, we replace the step size update strategy of TR-CMA-ES with the one from CMA-ES resulting in the "TR-CMA-ES with CSA" algorithm. For the second algorithm, we replace the TR-CMA-ES covariance update strategy with the one from CMA-ES resulting in the "TR-CMA-ES with fixed CMA coefficients" algorithm. The results in

**Algorithm 2** TR-CMA

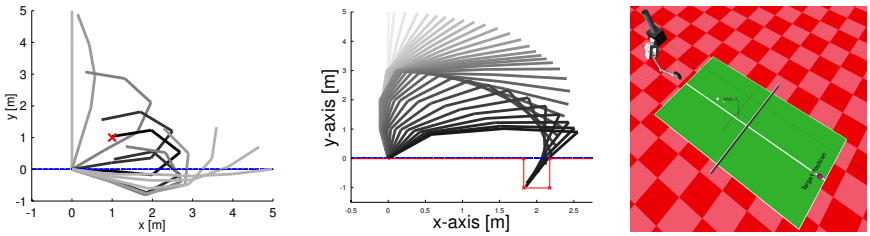
**given** $n$(Dimension), $N$(Number of Individuals)
**initialize** $\boldsymbol{\mu}_{t=0}, \sigma_{t=0} > 0, p_{c,t=0} = 0, \Sigma_{t=0} = \boldsymbol{I}, t = 0$
**repeat**
  **for** i = 1,...,$N$ **do**
    $\boldsymbol{x}_i = \boldsymbol{\mu}_t + \sigma_t \times \mathcal{N}(0, \boldsymbol{\Sigma}_t)$
    $f_i = f(\boldsymbol{x}_i)$
  **end for**
  $\boldsymbol{w}$ = **Compute Weights**$(\{\boldsymbol{x}_i, f_i\}_{i=1...N})$
  **Compute variance effective selection mass:** $\mu_w = \frac{\sum_{i=1}^{N} w_i}{\sum_{i=1}^{N} w_i^2}$
  **Set Hyper Parameters**
  $\lambda_{\boldsymbol{\Sigma}} = 1, \lambda_\sigma = \frac{2n}{(n+1.3)^2+\mu_w}, \epsilon_\Sigma = \min(0.2, 3\frac{\frac{\mu_w}{2}+\frac{1}{\mu_w}}{(n+3)^2+\mu_w})$
  $\epsilon_\sigma = 15\epsilon_\Sigma, \epsilon_{\boldsymbol{\mu}} = 1000\epsilon_\Sigma, c_c = \frac{\mu_w+2}{n+\mu_w+5}$
  **Update Mean:**
  $\eta^*_{\boldsymbol{\mu}} = \text{argmin}_\eta\, g_{\boldsymbol{\mu}}(\eta)$
  $\boldsymbol{\mu}_{t+1} = \frac{\eta^*_{\boldsymbol{\mu}}\boldsymbol{\mu}_t + \sum w_i \boldsymbol{x}_i}{\sum w_i + \eta^*_{\boldsymbol{\mu}}}$
  **Compute Evolution Path:**
  $\boldsymbol{p_c}_{t+1} = (1 - c_c^{1.4})\boldsymbol{p_c}_t + \sqrt{c_c(2-c_c)}\frac{\mu_w}{2}(\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_t)$
  **Update Covariance:**
  $\eta^*_{\boldsymbol{\Sigma}} = \text{argmin}_\eta\, g_{\boldsymbol{\Sigma}}(\eta)$
  $\boldsymbol{\Sigma}_{t+1} = \frac{\eta^*_{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}_t + \sum_i w_i \frac{(\boldsymbol{x}_i-\boldsymbol{\mu}_t)(\boldsymbol{x}_i-\boldsymbol{\mu}_t)^T}{\sigma_t} + \lambda_{\boldsymbol{\Sigma}} \frac{\boldsymbol{p_c}\boldsymbol{p_c}^T}{\sigma_t}}{\sum w_i + \lambda_{\boldsymbol{\Sigma}} + \eta^*_{\boldsymbol{\Sigma}}}$
  **Update Step-Size:**
  $\eta^*_\sigma = \text{argmin}_\eta\, g_\sigma(\eta)$
  $\sigma_{t+1} = \frac{n\eta^*_\sigma\sigma_t + \text{tr}(\boldsymbol{\Sigma}_t{}^{-1}(\sum_i w_i(\boldsymbol{x}_i-\boldsymbol{\mu}_t)(\boldsymbol{x}_i-\boldsymbol{\mu}_t)^T + \lambda_\sigma \boldsymbol{p_c}\boldsymbol{p_c}^T))}{n(\sum w_i + \lambda_\sigma + \eta^*_\sigma)}$
  $t = t + 1$
**until** stopping criterion is met

Figure 3.3(b) show that TR-CMA-ES outperforms all other algorithms while the two hybrid algorithms outperform CMA-ES. This result suggest that both step-size control and the KL-bound of TR-CMA-ES are effective improvements over CMA-ES.

**Small Sample-Sizes.** We furthermore optimize a 200 dimensional sphere function with sampling only 2 individuals in each iteration. The results in Figure 3.3(c) shows that while CMA-ES diverges, TR-CMA-ES robustly converges. CMA-ES doesn't use any regularizer on the mean update which results in fast divergence when size of population is very small w.r.t the problem dimension. CMA-ES could recover slightly by drastically decreasing the variance of the distribution, which again reduces the learning speed.

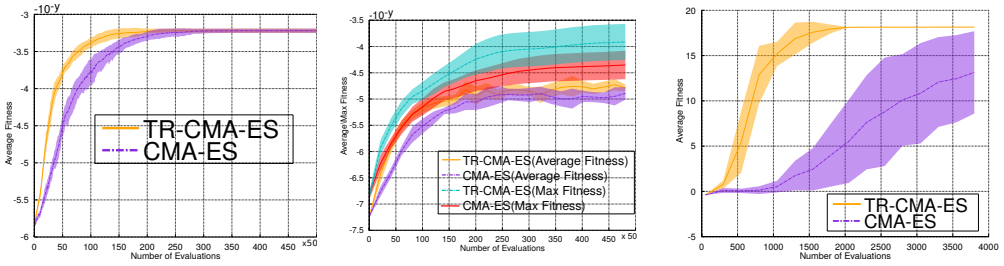
### 3.4.2 Simulated Robotics Tasks

We use a 5-link planar robot that has to reach a given point in task space as a toy task for the comparisons, see Figure 3.4(a). We subsequently made the task more difficult




(a) Reaching Task (b) Hole Reaching Task (c) Table Tennis

FIGURE 3.4: (a) Planar reaching task: a 5-link planar robot has to reach a via-point $\boldsymbol{v}_{50} = [1, 1]$ in task space. The via-point is indicated by the red cross. The postures of the resulting motion are shown as overlay, where darker postures indicate a posture which is close in time to the via-point. (b) Planar hole reaching task: A 5-link planar robot has to reach the bottom of a hole centring at point $[2\ 0]$ in task space while avoiding any collision with the walls. The hole is indicated by the red lines. The postures of the resulting motion are shown as overlay, where darker postures indicate a posture which is close in time to reach the bottom of the hole. (c) The table tennis task: The incoming ball has a fixed initial velocity. The goal of the robot is to learn forehand strokes to return the ball to a fixed target position.




(a) Reaching Task (b) Hole Reaching Task (c) Table Tennis

FIGURE 3.5: Comparisons for the three simulated robotics tasks which were used for comparison. (a) Reaching task (b) Hole reaching task and (c) Table Tennis task. Results show that TR-CMA-ES outperforms CMA-ES in terms of number of needed evaluations on all task. In the hole reaching task as well as in the table tennis task, TR-CMA-ES has a better average final performance than CMA-ES. Please note that y in $-10^{-y}$ is value on y axis.

by introducing hard obstacles, which results in a discontinuous fitness function. We denote this task as hole-reaching task, see Figure 3.4(b). Finally, we evaluate our algorithm on a physical simulation of a robot playing table tennis (Figure 3.4(c)). For all tasks, we use dynamic motor primitives (DMPs) [41] as trajectory generator and we optimise the parameters of the DMPs to achieve optimum movements. For all tasks, we generate 50 samples in each iterations. We ran 10 trials and we report the average fitness in each iteration and the standard deviations over all trials. All other settings are set as before if not stated otherwise.

**Planar Reaching Task.** For completing the reaching task, the robot has to reach a via-point $\boldsymbol{v}_{50} = [1, 1]$ at time step $50$ with its end-effector and at the final time step $T = 100$ the point $\boldsymbol{v}_{100} = [5, 0]$. The reward was given by a quadratic cost term for the distance from two via-points as well as quadratic costs for high accelerations. The DMPs goal attractor for reaching the final state was assumed to be known. Hence, the parameter vector $\boldsymbol{x}$ for a 5-link robot with 5 basis function for each degree of freedom had $25$ dimensions. Figure 3.5(c) shows the learning progress. TR-CMA-ES again outperform CMA-ES, while both algorithms could find the optimal solution.

**Planar Hole Reaching Task.** For completing the hole reaching task, the robot's end effector has to reach the bottom of a hole ($35cm$ wide and $1m$ deep) centered point $[2, 0]$ without any collision with the ground or the hole wall. The reward was given by a quadratic cost term for the distance to bottom of the hole, quadratic costs for high accelerations and quadratic costs for collisions with the environment. Note that this objective function is highly discontinuous due to the quadratic costs for the collisions. The DMP goal attractor for reaching the final state in this task is unknown and also need to be learned. Hence, our lower level policy for a 5-link robot with 5 basis functions for each degree of freedom had $30$ dimensions. Figure 3.5(b) shows the results. For this task we report both the maximum fitness and average fitness in each iteration. In both cases TR-CMA-ES illustrates a better performance.

**Table Tennis Task.** In this task, we use a simulated robot arm (see Figure 3.4(c)) to learn a forehand hitting stroke in a table tennis game. The robot is mounted on a floating base and has 8 actuated joints including the floating base. The goal of the robot is to return the incoming ball at a target position on the opponent's side of the table. The ball is always served with same initial velocities in the x,y and z directions. To learn the task, we initialize the mean of the distribution with a initial DMP trajectory obtained by kinesthetic teaching, such that the movement generates a single forehand stroke. We only learn the final positions and final velocities of the DMP trajectories as well as the $\tau$ time-scaling parameter and the starting time point of the DMP which results in 18 parameters. The reward function is defined by the sum of quadratic penalties for missing the ball (minimum distance between ball and racket trajectory) and missing the target return position. Figure 3.5(c) shows the result. TR-CMA-ES robustly finds the solution in all trials while CMA-ES could not find good solution in 3 trials out of 10. And TR-CMA-ES was always significantly faster.

## 3.5 Conclusion

In this chapter, we derived the full CMA-ES update equations for mean and covariance with an expectation-maximization based framework using information-geometric trust regions. The presented update for the covariance matrix share the same structure then the CMA-ES algorithm. However, CMA-ES is not using a trust region (i.e., use constraint on KL), but a penalty on KL is used in the objective as regularizer. As a consequence, the optimum 'Lagrangian' multipliers in CMA-ES are set by hand and remain fixed during the learning. However they are well established and can be left constant throughout many applications. In the trust region formulation, the Lagrangian multipliers are optimized for the given bound $\epsilon$. As we show, in addition to the theoretical foundation, this optimisation of the parameters gives us an improved performance over the original algorithm. Moreover, we also use the KL bound to obtain update rules for the mean and the step size parameters. CMA-ES does not use a regularizer for the mean update but directly use the unregularized maximum likelihood update. While the mean does not easily overfit for low-dimensional parameters spaces with a high number of individuals, an unregularized update is more problematic for high-dimensional parameter spaces where it might even diverge.

In contrast to the mean and the covariance update, our step-size update does not match the step-size update from CMA-ES. Both, the TR-CMA-ES and CMA-ES take advantage of evolution path to set step size $\sigma$. However our update rule for the step size is obtained from the same principle as used for the mean and the covariance matrix. Given the similarities in terms of the mean and covariance matrix update between CMA-ES and our algorithm, our step size control update rule is more consistent and a more principled than the update rule is used in standard CMA-ES. The new step-size update also performs favourably in our experiments and should also be preferred for CMA-ES due to the consistent derivation. Our algorithm also enjoys all the invariance properties of the CMA-ES.

For future work, we will investigate other theoretical aspects of our framework such as the theoretical difference between the forward KL and inverse KL trust region. We will also extend our framework for solving contextual stochastic search problems and full reinforcement learning problems.

# Chapter 4

# Contextual Covariance Matrix Adaptation Evolutionary Strategies

Many stochastic search algorithms are designed to optimize a fixed objective function to learn a task, i.e., if the objective function changes slightly, for example, due to a change in the situation or context of the task, relearning is required to adapt to the new context. For instance, if we want to learn a kicking movement for a soccer robot, we have to relearn the movement for different ball locations. Such relearning is undesired as it is highly inefficient and many applications require a fast adaptation to a new context/situation. Therefore, we investigate contextual stochastic search algorithms that can learn multiple, similar tasks simultaneously. Current contextual stochastic search methods are based on policy search algorithms and suffer from premature convergence and the need for parameter tuning. In this chapter, we extend the well known CMA-ES algorithm to the contextual setting and illustrate its performance on several contextual tasks. Our new algorithm, called contextual CMA-ES, leverage from contextual learning while it preserves all the features of standard CMA-ES such as stability, avoidance of premature convergence, step size control and a minimal amount of parameter tuning.

## 4.1 Introduction

The notion of multi-task learning[1] has been set down in the machine learning community for at least the past two decades [14]. The main motivation for contextual learning is the potential for exploiting relevant information available in related tasks by concurrent learning using a shared representation. Therefore, instead of learning one task at a time, we would like to learn multiple tasks at once and exploit the correlations between related tasks. We use a context vector to characterize a task which

[1]The terms "contextual learning" and "multi-task learning" are used interchangeably throughout this chapter.

typically changes from one task execution to the next. For example, consider a humanoid soccer robot that needs to pass the ball to its team mates which are positioned on different locations on the field. Here, the soccer robot should learn to kick the ball to any given target location, which is specified by the context vector, on the field. In such cases, learning for every possible context is clearly inefficient or even infeasible. Therefore our goal is to generalize learned tasks from similar contexts to a new context. To do so, we learn a context-dependent policy for a continuous range of contexts without restarting the learning process. In this chapter, we consider stochastic search algorithms for contextual learning. Stochastic search algorithms [32, 92, 83] optimize an objective function in a continuous domain. These algorithms assume that the objective function is a black box function, i.e., they only use the objective values and don't require gradients or higher-order derivatives of the objective function. Contextual stochastic search algorithms have been investigated in the field of policy search for robotics [55, 51]. However, these policy search algorithms typically suffer from premature convergence and perform unfavourably in comparison to state of the art stochastic search methods [90] such as the CMA-ES algorithm [32]. The CMA-ES algorithm is considered as the state of the art in stochastic optimization. CMA-ES performs favourably in many tasks without the need of extensive parameter tuning. The algorithm has many beneficial properties, including automatic step-size adaptation, efficient covariance updates that incorporates the current samples as well as the evolution path and its invariance properties. However, CMA-ES is lacking the important feature of contextual (multi-task) learning. Therefore, in this research, we extend the well known CMA-ES algorithm to contextual setting using inspiration from contextual policy search. Our new algorithm, called contextual CMA-ES, generalizes the learned solution to new, unseen contexts during the optimization process while it preserves all the features of standard CMA-ES such as stability, avoidance of premature convergence, step size control, a minimal amount of parameter tuning and simple implementation. In our derivation of the algorithm, we also provide a new theoretical justification for the covariance matrix update rule of CMA-ES algorithm that also applies to the non-contextual case and gives new insights into how the covariance update can be motivated. For illustration of the algorithm, we will use contextual standard functions and two contextual simulated robotic tasks which are robot table tennis, and a robot kick task. We show that our contextual CMA-ES algorithm performs favourably in comparison to other contextual learning algorithms.

## 4.2 Related Work

In order to generalize a learned parameter vector for a context to the other contexts, a standard approach is to optimize the parameters for several target contexts independently. Subsequently, regression methods are used to generalize the optimized contexts to a new, unseen context [16, 91]. Learning for multiple tasks without restarting the learning process is known as contextual (multi-task) policy search [55, 51, 21]. In the area of contextual stochastic search algorithms, such a multi-task learning capability was established for information-theoretic policy search algorithms [75], such as the Contextual Relative Entropy Policy Search (CREPS) algorithm [55]. However, it has been shown that REPS suffers from premature convergence [3, 5] as the update of the covariance matrix, which is based only on the current set of samples, is reducing the variance of the search distribution too quickly. In order to alleviate this problem, the authors of [3, 5] suggest to combine the sample covariance with the old covariance matrix, similar to the CMA-ES algorithm [32]. However this method does not take advantage of other features of CMA-ES such as step-size control. In [30] also, an evolutionary contextual learning algorithm for only single dimensional contextual problems was proposed. This method defines a set of discrete contexts and represents the mean of the search distribution of each segment of the space. They evolve the mean of each segment using the (1+1)-CMA-ES [40] algorithm. The used discretization inherently limits the approach to one dimensional context variable.

## 4.3 Preliminaries

In this section, we will first formulate the problem statement and subsequently explain contextual stochastic search algorithms in general. Afterwards, we will emphasize the contextual Relative Entropy Policy Search (REPS) algorithm [55], which will provide insights for the development of the new contextual CMA-ES algorithm.

### 4.3.1 Problem Statement

We consider contextual black-box optimization problems that are characterized by a $n_s$-dimensional context vector $\boldsymbol{s}$. The task is to find for each context vector $\boldsymbol{s}$, an optimal parameter vector $\boldsymbol{\theta}_{\boldsymbol{s}}^*$ that maximizes an objective function $\boldsymbol{R}_{\boldsymbol{s}}(\boldsymbol{\theta}) : \{\mathbb{R}^{n_s} \times \mathbb{R}^{n_\theta}\} \rightarrow \mathbb{R}$. Note that the objective function is also dependent on the given context vector $\boldsymbol{s}$. We want to find an optimal context dependent policy $m^*(\boldsymbol{s})$ in form of

$$\boldsymbol{\theta}_{\boldsymbol{s}}^* = m^*(\boldsymbol{s}) = \boldsymbol{A}^* \boldsymbol{\varphi}(\boldsymbol{s}), \quad m^*(\boldsymbol{s}) : \mathbb{R}^{n_s} \rightarrow \mathbb{R}^{n_{\boldsymbol{\theta}}}$$

that outputs the optimal parameter vector $\boldsymbol{\theta}_{\boldsymbol{s}}^{*}$ for the given context $\boldsymbol{s}$[2]. The vector $\boldsymbol{\varphi}(\boldsymbol{s})$ is an arbitrary $n_{\boldsymbol{\varphi}}$-dimensional feature function of the context $\boldsymbol{s}$ and the gain matrix $\boldsymbol{A}$ is a $n_{\boldsymbol{\theta}} \times n_{\boldsymbol{\varphi}}$ matrix that models the dependence of parameters $\boldsymbol{\theta}$ on the context $\boldsymbol{s}$. Throughout this chapter, we use $\boldsymbol{\varphi}(\boldsymbol{s}) = [1 \;\; \boldsymbol{s}]$, which results in linear generalization over contexts. However, other feature functions such as radial basis functions (RBF) for non-linear generalization over contexts [1] can also be used. The only accessible information on objective function $\boldsymbol{R}_{\boldsymbol{s}}(\boldsymbol{\theta})$ are returns $\{R_k\}_{k=1...N}$ of context-parameters samples $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k\}_{k=1...N}$, where $k$ is the index of the sample and $N$ is number of samples.

### 4.3.2 Contextual Stochastic Search

Contextual Stochastic search algorithms maintain a conditional search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ over the parameter space $\boldsymbol{\theta}$ of the objective function $\boldsymbol{R}_{\boldsymbol{s}}(\boldsymbol{\theta})$. The search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is often modeled as a linear Gaussian distribution, i.e.,

$$\pi(\boldsymbol{\theta}|\boldsymbol{s}) = \mathcal{N}\left(\boldsymbol{\theta}|m(\boldsymbol{s}) = \boldsymbol{A}\boldsymbol{\varphi}(\boldsymbol{s}), \sigma\boldsymbol{\Sigma}\right),$$

where $m(\boldsymbol{s})$ is a context dependent mean function that represents the context dependent policy we want to learn, $\boldsymbol{\Sigma}$ is a covariance matrix (shape of the distribution) and $\sigma$ is the step size (magnitude). Covariance matrix and step size are used for exploration and are independent of the context in most setups. In each iteration, $N$ context-parameter-return samples are generated with the current contextual policy. To do so, the context vectors $\boldsymbol{s}_k$ are drawn from a possibly unknown context distribution $\mu(\boldsymbol{s})$ [3]. Subsequently, the current search distribution $\pi^{t}(\boldsymbol{\theta}|\boldsymbol{s})$ is used to generate the parameter $\boldsymbol{\theta}_k$ for the corresponding context $\boldsymbol{s}_k$. For each sample $k$, the return $R_k$ of $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k\}$ is obtained by querying the objective function $\boldsymbol{R}_{\boldsymbol{s}}(\boldsymbol{\theta})$. Typically, the samples $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, R_k\}_{k=1...N}$ are used to compute a weight or pseudo probability $d_k$ for each sample $k$. Subsequently, using $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, d_k\}_{k=1...N}$, a new conditional Gaussian search distribution $\pi^{t+1}(\boldsymbol{\theta}|\boldsymbol{s})$ is estimated by updating the gain matrix $\boldsymbol{A}^{t+1}$ of the context-dependent mean function, the covariance matrix $\boldsymbol{\Sigma}^{t+1}$ and step size $\sigma^{t+1}$. This process is run iteratively until the algorithm converges to a solution. The final solution is the mean function $m^{*}(\boldsymbol{s})$ with the estimate of the optimal gain matrix $\boldsymbol{A}^{*}$ in the last iteration. Please note that, if the context vector is fixed, then the explained

[2] Please note that without loss of generality, in this research we learn a context dependent policy in this form. However the policy can be arbitrary complex such as neural network.

[3] Please note that the context samples depends on the task in an uncontrollable manner. However, throughout this chapter, we use a uniform distribution to sample contexts for simplicity.

algorithm reduces to standard stochastic search where the mean function is a constant. Algorithm 1 given in the supplement material shows a compact representation of contextual policy search methods[4].

### 4.3.3 Contextual REPS

Contextual REPS [55] is an instance of the general stochastic search algorithms introduced in the previous section where the weight computation and the distribution update are performed in a specific way.

**Computing the Weights.** In order to obtain new weights for the context-parameters-return samples in the data set, contextual REPS, optimizes for the joint probabilities $p(\boldsymbol{s}_k, \boldsymbol{\theta}_k)$. The key idea behind contextual REPS is to find a joint search distribution $p(\boldsymbol{s}, \boldsymbol{\theta})$ that maximizes the expected return i.e., $\max_p \iint p(\boldsymbol{s}, \boldsymbol{\theta}) R_s(\boldsymbol{\theta}) d\boldsymbol{s} d\boldsymbol{\theta}$, while it ensures a smooth and stable learning process by bounding the Kullback-Leibler divergence between the old search distribution $q$ and the newly estimated search distribution $p$, i.e, $\epsilon \geq \mathrm{KL}(p(\boldsymbol{s}, \boldsymbol{\theta}) || q(\boldsymbol{s}, \boldsymbol{\theta}))$. Please see [55] for full description of optimization program of contextual REPS. The solution of the contextual REPS optimization program results in a weight

$$d_k = \exp\left(\frac{R_k - V(\boldsymbol{s})}{\eta}\right) / Z \ , \ Z = \sum_{k=1}^{N} d_k, \tag{4.1}$$

for each context-parameter sample $[\boldsymbol{s}_k, \boldsymbol{\theta}_k]$. The function $V(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^T \boldsymbol{w}$ is a context-dependent baseline, similar to a value function, that depends linearly on features $\boldsymbol{\phi}(\boldsymbol{s})$ of the context vector $\boldsymbol{s}$. It is subtracted from the return $R$. Intuitively, this subtraction allows us to assess the quality of the samples independently from the experienced context. In this chapter, we use a quadratic feature function for the baseline, for example, in a one dimensional contextual problem, we use $\boldsymbol{\phi}(s) = [s, \ s^2]$. The parameters $\boldsymbol{w}$ and $\eta$ are Lagrangian multipliers that can be obtained by optimizing the convex dual function of the REPS optimisation program. The dual function is given in the supplement.

**Updating the Search Distribution.** In contextual REPS, the step size is fixed to 1, i.e., $\sigma^t = 1$. In order to obtain a new Gaussian search distribution $\pi^{t+1}$, contextual

[4] **Please download the supplementary file from** https://goo.gl/MLzKsW

REPS directly uses a weighted maximum likelihood estimate, i.e,

$$\underset{\boldsymbol{\Sigma}^{t+1},\boldsymbol{A}^{t+1}}{\operatorname{argmax}} \sum_{k=1}^{N} d_k \log \pi(\boldsymbol{\theta}_k|\boldsymbol{s}_k; \boldsymbol{\Sigma}^{t+1}, \boldsymbol{A}^{t+1}). \tag{4.2}$$

*Mean Function Update Rule.* We can efficiently solve the optimization program in equation 4.2 for $\boldsymbol{A}^{t+1}$ in closed form. The solution for $\boldsymbol{A}^{t+1}$ is given by

$$\boldsymbol{A}^{t+1} = \left(\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{U}, \tag{4.3}$$

where $\boldsymbol{\Phi}^T = [\boldsymbol{\varphi}_1, ..., \boldsymbol{\varphi}_N]$ contains the feature vector for the policy for all samples, $\boldsymbol{U} = [\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_N]$ contains all the sample parameters and $\boldsymbol{D}$ is the diagonal weighting matrix containing the weightings $d_k$. The term $\lambda \boldsymbol{I}$ is a regularization term.

*Covariance Matrix Update Rule.* We can also solve the optimization program in equation 4.2 for $\boldsymbol{\Sigma}^{t+1}$ in closed form. The solution for $\boldsymbol{\Sigma}^{t+1} = \boldsymbol{S}$ which is also known as sample covariance $\boldsymbol{S}$ matrix and is given by

$$\boldsymbol{S} = \sum_{k=1}^{N} d_k \left(\boldsymbol{\theta}_k - \boldsymbol{A}^{\boldsymbol{t+1}} \boldsymbol{\varphi}_k\right) \left(\boldsymbol{\theta}_k - \boldsymbol{A}^{\boldsymbol{t+1}} \boldsymbol{\varphi}_k\right)^T. \tag{4.4}$$

As contextual REPS, most contextual policy search algorithms only use the current set of samples to estimate the new search distribution. It has been already noted by several authors [5, 90] that such approach causes problems with premature convergence as the covariance matrix is overfitted to the data and, consequently, the covariance update reduces the variance too quickly.

## 4.4 Contextual Covariance Matrix Adaptation Evolutionary Strategies

Current contextual stochastic search algorithms are lacking the beneficial features from CMA-ES such as pre-mature convergence avoidance, step-size control and a minimal set of tuning parameters. To this end, we will contextualize the CMA-ES algorithm and hence inherit all its beneficial features. We will now explain the contextual CMA-ES rules for computing the weights as well as the distribution updates.

**Algorithm 3** Contextual CMA-ES

**given** $n,\ n_{\boldsymbol{s}},\ n_c = n + n_{\boldsymbol{s}}\ , N = 4 + 3\ln n_c(1 + 2n_{\boldsymbol{s}})$
**initialize** $\boldsymbol{A}^{t=0}, \sigma^{t=0} > 0, p_\sigma^{t=0} = 0, p_c^{t=0} = 0, \Sigma^{t=0} = \boldsymbol{I}, 0 \leftarrow t$
**repeat**
  **for** k = 1,...,$\lambda$ **do**
    **Observe** $\boldsymbol{s}_k$
    $m(\boldsymbol{s}_k) = \boldsymbol{A^t}\boldsymbol{\varphi}(\boldsymbol{s}_k)$
    $\boldsymbol{\theta}_k = m(\boldsymbol{s}_k) + \sigma^t \times \mathcal{N}\left(0, \boldsymbol{\Sigma}^t\right)$
    $R_k = \boldsymbol{R}_{\boldsymbol{s}_k}(\boldsymbol{\theta}_k)$
  **end for**
  $\boldsymbol{d}$ = **ComputeWeights**$(\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, R_k\}_{k=1...N})$ (Eq. 1 or 5)
  **Set Hyper Parameters:**
  $\mu_w = \frac{1}{\sum_{i=1}^N d_k^2}$(Number of effective samples)
  **Covariance Hyper Parameters**
  $c_1 = \frac{2\min(1,\lambda/6)}{(n_c+1.3)^2+\mu_w}, c_\mu = \frac{2(\mu_w - 2 + 1/\mu_w)}{(n_c+2)^2+\mu_w}, c_c = \frac{4}{4+n_c}$
  **Step Size Hyper Parameters**
  $c_\sigma = \frac{\mu_w+2}{n_c+\mu_w+3}, d_\sigma = 1 + c_\sigma + 2\sqrt{\frac{\mu_w-1}{n_c+1}} - 2 + \log(1 + 2n_{\boldsymbol{s}})$
  **Update Mean Function**(Eq. 3)
  **Update Evolution Path**
  $\hat{\boldsymbol{\varphi}} = \sum_{k=1}^N \frac{1}{N}\boldsymbol{\varphi}(\boldsymbol{s}_k)\ , \boldsymbol{y} = \frac{\boldsymbol{A^{t+1}}\hat{\boldsymbol{\varphi}} - \boldsymbol{A^t}^T\hat{\boldsymbol{\varphi}}}{\sigma^t}$
  $\boldsymbol{p}_c^{t+1} \leftarrow (1 - c_c)\boldsymbol{p}_c^t + h_\sigma\sqrt{c_c(2 - c_c)}\sqrt{\mu_w}\boldsymbol{y}$
  $\boldsymbol{p}_\sigma^{t+1} \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma^t + \sqrt{c_\sigma(2 - c_\sigma)}\sqrt{\mu_w}(\boldsymbol{\Sigma}^t)^{\frac{-1}{2}}\boldsymbol{y}$
  **Update Covariance Matrix**
  $\boldsymbol{S} = \frac{1}{(\sigma^t)^2}\sum_{k=1}^N d_k(\boldsymbol{\theta}_k - \boldsymbol{A^t}^T\boldsymbol{\varphi}(\boldsymbol{s}_k))(\boldsymbol{\theta}_k - \boldsymbol{A^t}^T\boldsymbol{\varphi}(\boldsymbol{s}_k))^T$
  $\boldsymbol{\Sigma}^{t+1} = (1 - c_1 - c_\mu)\boldsymbol{\Sigma}^t + \underbrace{c_\mu \boldsymbol{S}}_{\text{rank-}\mu\text{ update}} + \underbrace{c1\ \boldsymbol{p}_c^{t+1}\boldsymbol{p}_c^{t+1^T}}_{\text{rank-one update}}$
  **Update Step Size:** $\sigma^{t+1} \leftarrow \sigma^t \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{\mathbf{t+1}}\|}{\mathbb{E}\|\mathcal{N}(\mathbf{0},\boldsymbol{I})\|}\right)\right)$
  $t \leftarrow t + 1$
**until** stopping criterion is met

### 4.4.1 Computing the Weights

CMA-ES originally ranks the samples based on their returns and, subsequently, it weights the samples based on this ranking such that better samples get higher weights. However before we rank the samples, we need to correct the returns from their context-dependent part such that we can judge the quality of the parameter vector $\boldsymbol{\theta}$ independently of the quality of the context $\boldsymbol{s}$. To do so, inspired by the contextual REPS, we compute a context-dependent baseline $V(\boldsymbol{s})$ which we subtract from the returns $R_k$ to compute advantages $A_k$, i.e.,

$$A_k = R_k - V(\boldsymbol{s}_k).$$

The baseline function $V(\boldsymbol{s})$ is estimated from the current dataset $\{\boldsymbol{s}_k, R_k\}_{k=1...N}$ and captures the average return of the samples for context $\boldsymbol{s}$. I.e., $V(\boldsymbol{s})$ is a value

function that captures the expected return for a given context using the current search distribution. In order to learn the baseline, we can use ridge linear regression to fit a function of the form $V(\boldsymbol{s}) = \boldsymbol{\beta}^T \boldsymbol{\phi}(\boldsymbol{s})$, where $\boldsymbol{\phi}(\boldsymbol{s})$ defines the context features and $\boldsymbol{\beta}$ can be obtained using linear regression. In this chapter, the feature function $\boldsymbol{\phi}(\boldsymbol{s})$ is similarly defined as the feature function $\boldsymbol{\phi}$ used for contextual REPS, but it contains an additional bias term, i.e., $\boldsymbol{\phi}(\boldsymbol{s}) = [1 \quad \boldsymbol{\phi}(\boldsymbol{s})^T]^T$. After computing the new dataset $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, A_k\}_{k=1...N}$, we can now use the CMA-ES ranking to compute the weights $d_k$ fofootnoter each context-parameter sample pair $[\boldsymbol{s}_k, \boldsymbol{\theta}_k]$. We first sort the dataset $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, A_k\}_{k=1...N}$ in ascending order with respect to the advantage values $A_k$. Subsequently, the weight of the $jth$ best sample in the list is set to

$$d^j = \ln(N + 0.5) - \ln(j) \tag{4.5}$$

, which will give us the new dataset $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, d_k\}_{k=1...N}$ that will be used to update the search distribution. Without loss of generality, we will assume that the weights sum to 1.

### 4.4.2 Search Distribution Update Rule

Next, we will explain the update rules for contextual CMA-ES and give a new mathematical interpretation for covariance matrix update rule. We will explain the parts that are relevant for contextualizing the standard CMA-ES. For further explanations we refer to [31].

*Mean Function Update Rule.* In standard CMA-ES, the mean of the distribution is a constant and not a context-dependent function. To update the constant mean, standard CMA-ES uses weighted average of samples which is also the solution of weighted maximum likelihood estimate. Therefore, we directly use the update rule we obtained for contextual REPS from Equation 4.3 to obtain the mean function of our distribution.

*Covariance Matrix Update Rule.* The complete contextual CMA-ES algorithm including all parameter settings are outlined in Algorithm 1. We will refer to the lines of the Algorithm 1. The covariance matrix update of CMA-ES consists of two parts (Line 14) which are the rank-one and rank-$\mu$ updates. In standard CMA-ES, the rank-one update uses the evolution path vector of the consecutive updates of the mean $\boldsymbol{y} = \frac{\boldsymbol{m}^{t+1} - \boldsymbol{m}^t}{\sigma^t}$ of the search distribution. In contextual CMA-ES, the mean is now a context dependent function and not a constant. Therefore, to compute the evolution path $\boldsymbol{p}_c$ (Line 13), we use the expected update of the search distribution

over the context distribution $\mu^t(\boldsymbol{s})$, i.e,

$$\boldsymbol{y} = \frac{\mathbb{E}_{s\sim\mu^t(\boldsymbol{s})}[m^{t+1}(\boldsymbol{s}) - m^t(\boldsymbol{s})]}{\sigma^t}.$$

Computing $\boldsymbol{y}$ for our samples reads

$$\boldsymbol{y} = \frac{(\boldsymbol{A^{t+1}} - \boldsymbol{A^t})\hat{\boldsymbol{\varphi}}}{\sigma^t} \ , \ \hat{\boldsymbol{\varphi}} = \sum_{k=1}^{N} \frac{1}{N}\boldsymbol{\varphi}(\boldsymbol{s}_k). \tag{4.6}$$

Using the information from the evolution path leads to significant improvements in terms of convergence speed, as it enables the algorithm to exploit correlations between consecutive steps. For a full explanation see [31]. The rank-$\mu$ update in the standard CMA-ES algorithm incorporates the information about the current successful steps (Line 14). This information is stored in the sample covariance matrix $\boldsymbol{S}$ (Line 14). The sample covariance in the CMA-ES algorithm is computed differently than in REPS. While REPS uses the new context dependent mean function $\boldsymbol{m}^{t+1}(\boldsymbol{s})$ to compute the covariance matrix $\boldsymbol{\Sigma}^{t+1}$(Equation 4.4), CMA-ES uses the old context dependent mean function $\boldsymbol{m}^t(\boldsymbol{s})$, i.e.,

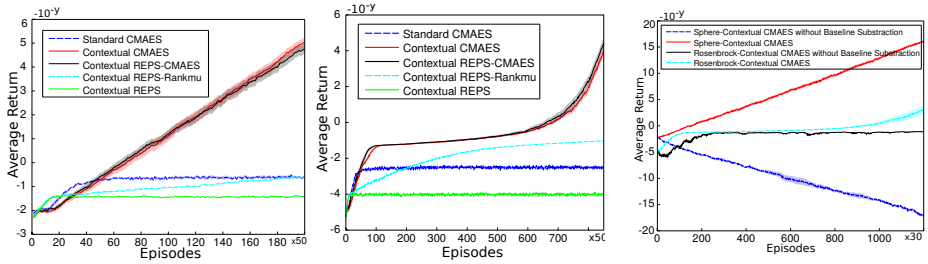
$$\boldsymbol{S}_{cma} = \frac{1}{\sigma^{t2}} \sum_{k=1}^{N} d_k \big(\boldsymbol{\theta}_k - \boldsymbol{A^t}\boldsymbol{\varphi}(\boldsymbol{s}_k)\big)\big(\boldsymbol{\theta}_k - \boldsymbol{A^t}\boldsymbol{\varphi}(\boldsymbol{s}_k)\big)^T.$$

By using the old mean function $m^t(\boldsymbol{s})$, we are increasing the likelihood of successful steps instead of likelihood of successful samples. This approach has been shown to be less prone to premature convergence [31]. The final covariance matrix update rule combines the old covariance matrix, sample covariance matrix and the evolution path information matrix, i.e,

$$\boldsymbol{\Sigma}^{t+1} = (1 - c_1 - c_\mu)\boldsymbol{\Sigma}^t + \underbrace{c_\mu \boldsymbol{S}_{cma}}_{\text{rank-}\mu\text{ update}} + \underbrace{c_1 \, \boldsymbol{p}_c^{t+1}\boldsymbol{p}_c^{t+1^T}}_{\text{rank-one update}}.$$

The factors $c_1$ and $c_\mu$ are the corresponding factors for rank-one and rank-$\mu$ updates such that $c_1 + c_\mu \leq 1$.

**Interpretation of the CMA-ES Covariance Update Rule.** Originally, the CMA-ES update rules have been obtained from intuitive, well-defined heuristics. Recently, it has been shown that the rank-$\mu$ update of covariance follows an approximate natural gradient of the expected returns [6]. In this section, we give a new mathematical interpretation for contextual CMA-ES covariance matrix update rule which also applies to standard CMA-ES. The covariance update rule in Line 14 has been shown




(a) Sphere (b) Rosenbrock (c) Baseline evaluation

FIGURE 4.1: The performance comparison of stochastic search methods for optimizing contextual version of standard functions (a)Sphere (b)Rosenbrock, The results show that while both contextual CMA-ES and contextual REPS-CMAES perform well, Contextual REPS suffers from premature convergence and contextual REPS-rank$\mu$ is very slow which shows the importance of step size control and incorporation of evolution path. We also compared with standard CMA-ES to show the importance of contextual version of CMA-ES. (c) Evaluation of influence of baseline in contextual CMA-ES for the Sphere and Rosenbrock functions. As the figure shows, the baseline is crucial for good performance of contextual CMA-ES. Please note that y in $-10^{-y}$ is value on y axis.

very effective for reproducing past successful steps while avoiding premature convergence. In fact, this update rule can be obtained by maximizing the likelihood of weighted steps as well as the weighted evolution path-step while minimizing the KL-divergence between new and old search distribution to avoid over-fitting and premature convergence, i.e,

$$\underset{\boldsymbol{\Sigma}^{t+1}|m=m^t}{\operatorname{argmax}} \underbrace{\sum_{k=1}^{N} d_k \log \pi^{t+1}(\boldsymbol{\theta}_k|\boldsymbol{s}_k)}_{\text{successful steps}} + \underbrace{\lambda \log \pi^{t+1}(\boldsymbol{p}_c^{t+1} + m^t(\hat{\boldsymbol{s}})|\hat{\boldsymbol{s}})}_{\text{evolution path}} - \underbrace{\gamma \mathrm{KL}(\pi^t|\pi^{t+1})}_{\text{Avoids overfitting}}.$$

Where $\hat{\boldsymbol{s}} = \sum_{k=1}^{N} \frac{1}{N} \boldsymbol{s}_k$. The notation $\{\boldsymbol{\Sigma}^{t+1}|m = m^t\}$ means that we optimize for $\boldsymbol{\Sigma}^{t+1}$ while the mean function is set to the old mean function $m^t$ which results in an optimized $\boldsymbol{\Sigma}^{t+1}$ for successful steps. $\lambda > 0$ and $\gamma > 0$ define the trade off between maximizing the likelihood of successful steps and keeping the KL-divergence of the new and old search distribution small. By setting $\lambda$ and $\gamma$ to zero and setting the mean function to the new one i.e., $m = m^{t+1}$, we obtain the sample covariance matrix $\boldsymbol{S}$ used by REPS. Considering a Gaussian search distribution, we can solve this optimization program in closed form and obtain the exact form of covariance matrix update rule as shown in Line 14 (Please see supplement material for derivation details). This derivation allows for the first time to formulate a clearly defined objective

to obtain the full CMAES update rules which gives us a better understanding of the algorithm.

### 4.4.3 Step Size Update Rule

The step-size adaptation control of CMAES also uses the evolution path vector $\boldsymbol{p}_\sigma$ (line 13) to correct the step size. The intuition is that if the successful steps between consecutive search distributions are towards the same direction, i.e., they are correlated, the updates will sum up and the evolution path will have a high magnitude. In this case, the step size should be increased. If the update directions cancel each other out, the evolution path will have a small magnitude and the step size should be decreased. In the contextual case, similar to the rank-one update, we use the Equation 4.6 to compute the mean update between two consecutive search distribution. Line 15 shows the step size update rule. For a full explanation about CMA-ES step size control, see [31].

## 4.5 Experiments

In this section, we evaluate four contextual algorithms such as CMA-ES, REPS, REPS-CMAES and REPS-Rank$\mu$ [3][5]. The REPS-CMAES algorithm uses the weighting method of REPS and the distribution update rule of CMA-ES. REPS-Rank$\mu$ also uses the weighting method of REPS but it uses only the rank-$\mu$ update rule of CMA-ES without step size adaptation. Please see the supplement for more details on differences of these algorithms. We also evaluate simultaneous multi task learning versus learning tasks in isolation. We chose two series of optimization tasks for comparisons. In the first series, we use the standard optimization test functions [67] , and for the second series of optimization tasks, we use a robot table tennis task. For all experiments, the KL-bound $\epsilon$ for REPS is set to 1 and for all experiments we use the default hyper parameter settings given in Algorithm 1 without further tuning. Please note that if the context dimension $n_s$ is set to zero, we get the parameter setting for standard CMA-ES.

### 4.5.1 Standard Functions

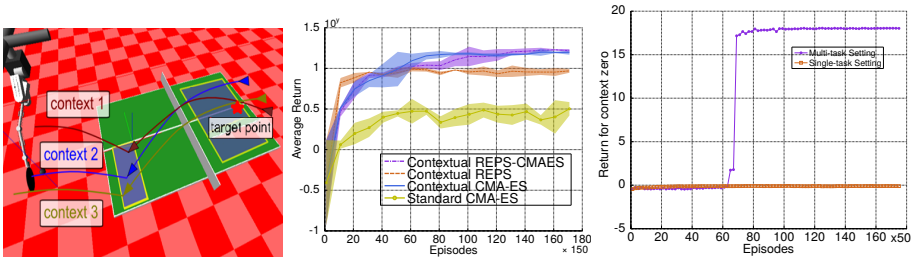
We chose two standard optimization functions which are the Sphere function $R_{\boldsymbol{s}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \boldsymbol{x}_i^2$, and the Rosenbrock function $R_{\boldsymbol{s}}(\boldsymbol{\theta}) = \sum_{i=1}^{n-1}[100(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i^2)^2 + (1 - \boldsymbol{x}_i)^2]$,

[5] **Matlab scripts of proposed algorithm and for reproducing the results on standard functions as well as videos regarding the experiments (table tennis and robot kick) and supplementary file are available at** https://goo.gl/MLzKsW

where $\boldsymbol{x} = \boldsymbol{\theta} + \boldsymbol{G}\boldsymbol{s}$ adapts the sample $\boldsymbol{\theta}$ linearly with the context. The matrix $\boldsymbol{G}$ is a constant matrix and is sampled from a normal distribution. Our standard functions have a global minimum with a value of zero for every context. However our algorithm here is a maximizer. Therefore, we multiply the functions by -1 such that now their maximum is 0. We want to find an optimum policy that for every context $\boldsymbol{s}$ outputs the optimal parameter $\boldsymbol{\theta}^*$. The optimum $\boldsymbol{\theta}_s^*$ for these functions is linearly dependent on the given context $\boldsymbol{s}$, hence, we initially test the performance of the algorithms under 'ideal contextual' conditions, i.e., the contextual policy is able to represent the optimal parameter vector $\boldsymbol{\theta}_{\boldsymbol{s}}^*$ for each context $\boldsymbol{s}$. We choose a linear policy $m(\boldsymbol{s}) = \boldsymbol{A}\boldsymbol{s} + b$. For the initial linear policy, $\boldsymbol{A}$ matrix is set to zero and b is sampled from a normal distribution. The initial covariance matrix is the identity matrix and the initial step size $\sigma$ is 1. Moreover the contexts are sampled uniformly from interval $1 \leq \boldsymbol{s}_i \leq 2, i = 1, \ldots, n_s$ where $n_s$ is dimension of the context space $\boldsymbol{s}$. For the Sphere we used a two dimensional context and 20 dimensional parameters while for the Rosenbrock function, we used a one dimensional context vector and 20 dimensional parameters. In each iteration we generate 50 context-parameters samples. We perform 20 trials for each experiment. And We report the average return of context-parameters samples in each iteration and the standard deviation over all 20 trials.

**Algorithmic Comparison.**

We compared contextual CMA-ES, contextual REPS, contextual REPS-CMAES, contextual REPS-Rank-$\mu$ and standard CMA-ES. The results in Figure 7.3(a) and Figure 7.3(b) show that contextual CMA-ES and contextual REPS-CMAES could successfully learn the contextual tasks while contextual REPS suffers from premature convergence and REPS-rank$\mu$ is too slow. As REPS-rank$\mu$ does not have step size control, this result shows the importance of step size control. Standard CMA-ES could not learn the task as it does not have any knowledge of the context. Please note that in this task standard CMA-ES which is a non-contextual algorithm has better performance than the contextual REPS. The reason is that contextual REPS suffers from premature convergence due to using only sample covariance. We already used more samples for contextual REPS to reduce the variance of covariance estimate and found a setting where contextual REPS found a better policy than standard CMA-ES. Yet, contextual REPS, needs much more samples to find such a policy. However contextual REPS with a proper covariance adaptation performs favourably(See Contextual REPS-CMAES).




(a) Robot Table Tennis (b) Table tennis algorithm comparison (c) Multi-task setup VS single-task setup

FIGURE 4.2: (a)The table tennis learning setup. The incoming ball has different initial velocity which is used as context vector. The goal of the robot is to learn forehand strokes to return the ball to a fixed target position.(b) Comparison of contextual algorithms on the table tennis task. Contextual REPS-CMAES achieves slightly better final performance. Please note that y in $10^y$ is value on y axis. (c) We trained the robot for a single, but hard context when the ball bounces at the middle of the table in the x-axis(red trajectory in Figure (a)) . In this case, the required solution is quite different from the initial solution. Due to the difficulty of the task, the robot could not learn the task and only found a locally optimal solution that hits the ball, but could not place it on the other side of the table. We also trained the robot in contextual setting where the context range includes the desired context but also easier tasks. The results show that the robot can learn even the complex task in this contextual setting as the easier tasks provide a guidance to the correct solution also for the difficult task.

**Evaluation of the Baseline.**

We also evaluated the influence of the baseline term that we use for contextual CMA-ES weighting. We use the sphere function with 3 dimensional context and 20 parameters. We also use the Rosenbrock with 1 dimensional context and 20 parameters. We generate 30 samples in each iteration. The results in Figure 7.3(c) show that without baseline term, i.e., $V(\boldsymbol{s}) = 0$, contextual CMA-ES can not find a good solution. Therefore, the baseline is a crucial part of the algorithm.

### 4.5.2 Robot Table Tennis

In this task, we use a simulated robot arm (see Figure 4.2(a)) to learn forehand hitting strokes in a table tennis game. The robot is mounted on a floating base and has 8 actuated joints including the floating base. The goal of the robot is to return the incoming ball at a target position on the opponent's side of the table. However, the ball is always served differently towards the robot with different initial velocities in the x-direction. We use the initial velocity of the ball as context, i.e., $\boldsymbol{s} = [v_x]$. To learn the task we use a linear policy $m(\boldsymbol{s}) = \boldsymbol{A}\boldsymbol{s} + b$ which is also the mean function

of distribution. We initialize $b$ with a initial DMP trajectory obtained by kinesthetic teaching, such that the movement generates a single forehand stroke. Other parameters have the same initialization as we did for standard functions. We only learn the final positions and final velocities of the DMP trajectories as well as the $\tau$ time-scaling parameter and the starting time point of the DMP which results in 18 parameters vector $\boldsymbol{\theta}$. The reward function is defined by the sum of quadratic penalties for missing the ball (minimum distance between ball and racket trajectory) and missing the target return position.

#### Algorithm Comparison

We compared contextual stochastic search methods on table tennis task. The results in Figure 4.2(b) shows that both the contextual CMA-ES and contextual REPS-CMA-ES can learn the task. However REPS-CMAES slightly outperforms contextual CMA-ES with the final solution. Contextual REPS again suffers from pre mature convergence. We also see that standard CMA-ES fails to learn this task.

#### Multi task learning versus Single task learning

In this experiment, we want to show that multi task learning can even facilitate learning of a single task. To do so, we choose a hard context to learn as it is shown in Figure 4.2(a) with a red trajectory. Here, the ball is served directly towards the robot and it lands close to the border of the table, which requires a quite different movement as the initial solution. We use the standard CMA-ES algorithm to learn this task, but as the results in Figure 4.2(c) show, the algorithm failed to learn it. However, when we use contextual CMA-ES with a context range that includes this desired context but also simpler tasks, it manages to learn this task 4.2(c). Hence, the simpler tasks guided the algorithm to find also a good solution for the hard task and avoid the local minimum found by the single task learner.

## 4.6 Conclusion and Future Work

Stochastic search methods such as CMA-ES have been employed extensively for black box optimization. However, these algorithms lack the important feature of contextual learning. Therefore we extended CMA-ES for contextual setting while we also provide a new theoretical justification for its covariance update rule. It turns out using baseline, the old covariance matrix and the step size control are crucial ingredients for a competitive performance. One interesting observation is that contextual learning also facilitates learning single tasks. The reason is that easier tasks

can guide the optimisation for learning harder tasks. For the future work we will investigate the application of the contextual CMA-ES for full reinforcement learning problems where we need to find the optimal actions for task states.

# Chapter 5

# Model-Based Relative Entropy Stochastic Search

Stochastic search algorithms are general black-box optimizers. Due to their ease of use and their generality, they have recently also gained a lot of attention in operations research, machine learning and policy search. Yet, these algorithms require a lot of evaluations of the objective, scale poorly with the problem dimension, are affected by highly noisy objective functions and may converge prematurely. To alleviate these problems, we introduce a new surrogate-based stochastic search approach. We learn simple, quadratic surrogate models of the objective function. As the quality of such a quadratic approximation is limited, we do not greedily exploit the learned models. The algorithm can be misled by an inaccurate optimum introduced by the surrogate. Instead, we use information theoretic constraints to bound the 'distance' between the new and old data distribution while maximizing the objective function. Additionally the new method is able to sustain the exploration of the search distribution to avoid premature convergence. We compare our method with state of art black-box optimization methods on standard uni-modal and multi-modal optimization functions, on simulated planar robot tasks and a complex robot ball throwing task. The proposed method considerably outperforms the existing approaches.

## 5.1 Introduction

Stochastic search algorithms [32, 92, 90, 83] are black box optimizers of an objective function that is either unknown or too complex to be modelled explicitly. These algorithms only make weak assumption on the structure of underlying objective function. They only use the objective values and don't require gradients or higher derivatives of the objective function. Therefore, they are well suited for black box optimization problems. Stochastic search algorithms typically maintain a stochastic search distribution over parameters of the objective function, which is typically a multivariate Gaussian distribution [32, 92, 90]. This policy is used to create samples from the

objective function. Subsequently, a new stochastic search distribution is computed by either computing gradient based updates [92, 83, 27], evolutionary strategies [32], the cross-entropy method [61], path integrals [90, 94], or information-theoretic policy updates [55]. Information-theoretic policy updates [75, 55, 92] bound the relative entropy (also called Kullback Leibler or KL divergence) between two subsequent policies. Using a KL-bound for the update of the search distribution is a common approach in the stochastic search. However, such information theoretic bounds could so far only be approximately applied either by using Taylor-expansions of the KL-divergence resulting in natural evolutionary strategies (NES) [92, 98], or sample-based approximations, resulting in the relative entropy policy search (REPS) [55] algorithm. In this chapter, we present a novel stochastic search algorithm which is called MOdel-based Relative-Entropy stochastic search (MORE). For the first time, our algorithm bounds the KL divergence of the new and old search distribution in closed form without approximations. We show that this exact bound performs considerably better than approximated KL bounds.

In order to do so, we locally learn a simple, quadratic surrogate of the objective function. The quadratic surrogate allows us to compute the new search distribution analytically where the KL divergence of the new and old distribution is bounded. Therefore, we only exploit the surrogate model locally which prevents the algorithm to be misled by inaccurate optima introduced by an inaccurate surrogate model.

However, learning quadratic reward models directly in parameter space comes with the burden of quadratically many parameters that need to be estimated. We therefore investigate new methods that rely on dimensionality reduction for learning such surrogate models. In order to avoid over-fitting, we use a supervised Bayesian dimensionality reduction approach. This dimensionality reduction technique avoids over fitting, which makes the algorithm applicable also to high dimensional problems. In addition to solving the search distribution update in closed form, we also upper-bound the entropy of the new search distribution to ensure that exploration is sustained in the search distribution throughout the learning progress, and, hence, premature convergence is avoided. We will show that this method is more effective than commonly used heuristics that also enforce exploration, for example, adding a small diagonal matrix to the estimated covariance matrix [90].

We provide a comparison of stochastic search algorithms on standard objective functions used for benchmarking and in simulated robotics tasks. The results show that MORE considerably outperforms state-of-the-art methods.

### 5.1.1 Problem Statement

We want to maximize an objective function $R(\boldsymbol{\theta}) : \mathbb{R}^n \rightarrow \mathbb{R}$. The goal is to find one or more parameter vectors $\boldsymbol{\theta} \in \mathbb{R}^n$ which have the highest possible objective value. We maintain a search distribution $\pi(\boldsymbol{\theta})$ over the parameter space $\boldsymbol{\theta}$ of the objective function $R(\boldsymbol{\theta})$. The search distribution $\pi(\boldsymbol{\theta})$ is implemented as a multivariate Gaussian distribution, i.e., $\pi(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. In each iteration, the search distribution $\pi(\boldsymbol{\theta})$ is used to create samples $\boldsymbol{\theta}^{[k]}$ of the parameter vector $\boldsymbol{\theta}$. Subsequently, the (possibly noisy) evaluation $R^{[k]}$ of $\boldsymbol{\theta}^{[k]}$ is obtained by querying the objective function. The samples $\{\boldsymbol{\theta}^{[k]}, R^{[k]}\}_{k=1...N}$ are subsequently used to compute a new search distribution. This process will run iteratively until the algorithm converges to a solution.

### 5.1.2 Related Work

Recent information-theoretic (IT) policy search algorithms [55] are based on the relative entropy policy search (REPS) algorithm which was proposed in [75] as a step-based policy search algorithm. However, in [55] an episode-based version of REPS that is equivalent to stochastic search was presented. The key idea behind episode-based REPS is to control the exploration-exploitation trade-off by bounding the relative entropy between the old 'data' distribution $q(\boldsymbol{\theta})$ and the newly estimated search distribution $\pi(\boldsymbol{\theta})$ by a factor $\epsilon$. Due to the relative entropy bound, the algorithm achieves a smooth and stable learning process. However, the episodic REPS algorithm uses a sample based approximation of the KL-bound, which needs a lot of samples in order to be accurate. Moreover, a typical problem of REPS is that the entropy of the search distribution decreases too quickly, resulting in premature convergence.

Taylor approximations of the KL-divergence have also been used very successfully in the area of stochastic search, resulting in natural evolutionary strategies (NES). NES uses the natural gradient to optimize the objective [92]. The natural gradient has been shown to outperform the standard gradient in many applications in machine learning [7]. The intuition of the natural gradient is that we want to obtain an update direction of the parameters of the search distribution that is most similar to the standard gradient while the KL-divergence between new and old search distributions is bounded. To obtain this update direction, a second order approximation of the KL, which is equivalent to the Fisher information matrix, is used.

Surrogate based stochastic search algorithms [58][59] have been shown to be more sample efficient than direct stochastic search methods and can also smooth out the noise of the objective function. For example, an individual optimization method is used on the surrogate that is stopped whenever the KL-divergence between the new

and the old distribution exceeds a certain bound [58]. For the first time, our algorithm uses the surrogate model to compute the new search distribution analytically, which bounds the KL divergence of the new and old search distribution, in closed form.

Quadratic models have been used successfully in trust region methods for local surrogate approximation [78, 77]. These methods do not maintain a stochastic search distribution but a point estimate and a trust region around this point. They update the point estimate by optimizing the surrogate and staying in the trusted region. Subsequently, heuristics are used to increase or decrease the trusted region. In the MORE algorithm, the trusted region is defined implicitly by the KL-bound.

The Covariance Matrix Adaptation-Evolutionary Strategy (CMA-ES) is considered as the state of the art in stochastic search optimization. CMA-ES also maintains a Gaussian distribution over the problem parameter vector and uses well-defined heuristics to update the search distribution.

## 5.2 Model-Based Relative Entropy Stochastic Search

Similar to information theoretic policy search algorithms [55], we want to control the exploration-exploitation trade-off by bounding the relative entropy of two subsequent search distribution. However, by bounding the KL, the algorithm can adapt the mean and the variance of the algorithm. In order to maximize the objective for the immediate iteration, the shrinkage in the variance typically dominates the contribution to the KL-divergence, which often leads to a premature convergence of these algorithms. Hence, in addition to control the KL-divergence of the update, we also need to control the shrinkage of the covariance matrix. Such a control mechanism can be implemented by lower-bounding the entropy of the new distribution. In this research, we will set the bound always to a certain percentage of the entropy of the old search distribution such that MORE converges asymptotically to a point estimate.

### 5.2.1 The MORE framework

Similar as in [55], we can formulate an optimization problem to obtain a new search distribution that maximizes the expected objective value while upper-bounding the KL-divergence and lower-bounding the entropy of the distribution

$$\max_{\pi} \int \pi(\boldsymbol{\theta}) \mathcal{R}_{\boldsymbol{\theta}} d\boldsymbol{\theta}, \quad \text{s.t. } \mathrm{KL}\big(\pi(\boldsymbol{\theta}) || q(\boldsymbol{\theta})\big) \leq \epsilon, \qquad H(\pi) \geq \beta, \qquad 1 = \int \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{5.1}$$

where $\mathcal{R}_{\boldsymbol{\theta}}$ denotes the expected objective[1] when evaluating parameter vector $\boldsymbol{\theta}$. The term $H(\pi) = -\int \pi(\boldsymbol{\theta}) \log \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}$ denotes the entropy of the distribution $\pi$ and $q(\theta)$ is the old distribution. The parameters $\epsilon$ and $\beta$ are user-specified parameters to control the exploration-exploitation trade-off of the algorithm.

We can obtain a closed form solution for $\pi(\boldsymbol{\theta})$ by optimizing the Lagrangian for the optimization problem given above. This solution is given as

$$\pi(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta})^{\eta/(\eta+\omega)} \exp\left(\frac{\mathcal{R}_{\boldsymbol{\theta}}}{\eta+\omega}\right), \tag{5.2}$$

where $\eta$ and $\omega$ are the Lagrangian multipliers. As we can see, the new distribution is now a geometric average between the old sampling distribution $q(\boldsymbol{\theta})$ and the exponential transformation of the objective function. Note that, by setting $\omega = 0$, we obtain the standard episodic REPS formulation [55]. The optimal value for $\eta$ and $\omega$ can be obtained by minimizing the dual function $g(\eta, \omega)$ such that $\eta > 0$ and $\omega > 0$, see [12]. The dual function $g(\eta, \omega)$ is given by

$$g(\eta, \omega) = \eta\epsilon - \omega\beta + (\eta+\omega) \log\left(\int q(\boldsymbol{\theta})^{\frac{\eta}{\eta+\omega}} \exp\left(\frac{\mathcal{R}_{\boldsymbol{\theta}}}{\eta+\omega}\right) d\boldsymbol{\theta}\right). \tag{5.3}$$

As we are dealing with continuous distributions, the entropy can also be negative. We specify $\beta$ such that the relative difference of $H(\pi)$ to a minimum exploration policy $H(\pi_0)$ is decreased for a certain percentage, i.e., we change the entropy constraint to

$$H(\pi) - H(\pi_0) \geq \gamma(H(q) - H(\pi_0)) \Rightarrow \beta = \gamma(H(q) - H(\pi_0)) + H(\pi_0).$$

Throughout all our experiments, we use the same $\gamma$ value of $0.99$ and we set minimum entropy $H(\pi_0)$ of search distribution to a small enough value like $-75$. We will show that using the additional entropy bound considerably alleviates the premature convergence problem.

### 5.2.2 Analytic Solution of the Dual-Function and the Policy

Using a quadratic surrogate model of the objective function, we can compute the integrals in the dual function analytically, and, hence, we can satisfy the introduced bounds exactly in the MORE framework. At the same time, we take advantage of surrogate models such as a smoothed estimate in the case of noisy objective functions and a decrease in the sample complexity[2].

[1] Note that we are typically not able to obtain the expected reward but only a noisy estimate of the underlying reward distribution.

[2] The regression performed for learning the quadratic surrogate model estimates the expectation of the objective function from the observed samples.

We will for now assume that we are given a quadratic surrogate model

$$\mathcal{R}_{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^T \boldsymbol{R} \boldsymbol{\theta} + \boldsymbol{\theta}^T \boldsymbol{r} + r_0$$

of the objective function $\mathcal{R}_{\boldsymbol{\theta}}$ which we will learn from data in Section 5.3. Moreover, the search distribution is Gaussian, i.e., $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{b}, \boldsymbol{Q})$. In this case the integrals in the dual function given in Equation 5.3 can be solved in closed form. The integral inside the log-term in Equation (5.3) now represents an integral over an un-normalized Gaussian distribution. Hence, the integral evaluates to the inverse of the normalization factor of the corresponding Gaussian. After rearranging terms, the dual can be written as

$$g(\eta, \omega) = \eta\epsilon - \beta\omega + \frac{1}{2}\left(\boldsymbol{f}^T \boldsymbol{F} \boldsymbol{f} - \eta \boldsymbol{b}^T \boldsymbol{Q}^{-1} \boldsymbol{b} - \eta \log |2\pi \boldsymbol{Q}| + (\eta + \omega) \log |2\pi(\eta + \omega)\boldsymbol{F}|\right) \tag{5.4}$$

with $\boldsymbol{F} = (\eta \boldsymbol{Q}^{-1} - 2\boldsymbol{R})^{-1}$ and $\boldsymbol{f} = \eta \boldsymbol{Q}^{-1} \boldsymbol{b} + \boldsymbol{r}$. Hence, the dual function $g(\eta, \omega)$ can be efficiently evaluated by matrix inversions and matrix products. Note that, for a large enough value of $\eta$, the matrix $\boldsymbol{F}$ will be positive definite and hence invertible even if $\boldsymbol{R}$ is not. In our optimization, we always restrict the $\eta$ values such that $\boldsymbol{F}$ stays positive definite[3].

Nevertheless, we could always find the $\eta$ value with the correct KL-divergence. In contrast to MORE, Episodic REPS relies on a sample based approximation of the integrals in the dual function in Equation (5.3). It uses the sampled rewards $\mathcal{R}_{\boldsymbol{\theta}}$ of the parameters $\theta$ to approximate this integral.

We can also obtain the update rule for the new policy $\pi(\boldsymbol{\theta})$. From Equation (5.2), we know that the new policy is the geometric average of the Gaussian sampling distribution $q(\boldsymbol{\theta})$ and a squared exponential given by the exponentially transformed surrogate. After re-arranging terms and completing the square, the new policy can be written as

$$\pi(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{F}\boldsymbol{f}, \boldsymbol{F}(\eta + \omega)\right), \tag{5.5}$$

where $\boldsymbol{F}$, $\boldsymbol{f}$ are given in the previous section.

## 5.3 Learning Approximate Quadratic Models

In this section, we show how to learn a quadratic surrogate. Note that we use the quadratic surrogate in each iteration to locally approximate the objective function and not globally. As the search distribution will shrink in each iteration, the model

[3] To optimize $g$, any constrained nonlinear optimization method can be used[13].

error will also vanish asymptotically. A quadratic surrogate is also a natural choice if a Gaussian distribution is used, cause the exponent of the Gaussian is also quadratic in the parameters. Hence, even using a more complex surrogate, it can not be exploited by a Gaussian distribution. A local quadratic surrogate model provides similar second-order information as the Hessian in standard gradient updates. However, a quadratic surrogate model also has quadratically many parameters which we have to estimate from a (ideally) very small data set. Therefore, already learning a simple local quadratic surrogate is a challenging task. In order to learn the local quadratic surrogate, we can use linear regression to fit a function of the form $f(\boldsymbol{\theta}) = \boldsymbol{\phi}(\boldsymbol{\theta})\boldsymbol{\beta}$, where $\boldsymbol{\phi}(\boldsymbol{\theta})$ is a feature function that returns a bias term, all linear and all quadratic terms of $\boldsymbol{\theta}$. Hence, the dimensionality of $\boldsymbol{\phi}(\boldsymbol{\theta})$ is $D = 1 + d + d(d+1)/2$, where $d$ is the dimensionality of the parameter space. To reduce the dimensionality of the regression problem, we project $\boldsymbol{\theta}$ in a lower dimensional space $l_{p\times 1} = \boldsymbol{W}\boldsymbol{\theta}$ and solve the linear regression problem in this reduced space[4]. The quadratic form of the objective function can then be computed from $\boldsymbol{\beta}$ and $\boldsymbol{W}$. Still, the question remains how to choose the projection matrix $\boldsymbol{W}$. We did not achieve good performance with standard PCA [43] as PCA is unsupervised. Yet, the $\boldsymbol{W}$ matrix is typically quite high dimensional such that it is hard to obtain the matrix by supervised learning and simultaneously avoid over-fitting. Inspired by [62], where supervised Bayesian dimensionality reduction are used for classification, we also use a supervised Bayesian approach where we integrate out the projection matrix $\boldsymbol{W}$.

### 5.3.1 Bayesian Dimensionality Reduction for Quadratic Functions

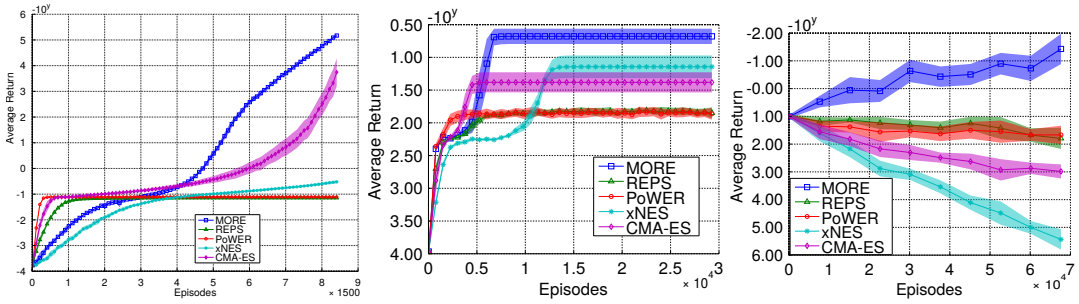
In order to integrate out the parameters $\boldsymbol{W}$, we use the following probabilistic dimensionality reduction model

$$p(r_*|\boldsymbol{\theta}_*, \boldsymbol{D}) = \int p(r_*|\boldsymbol{\theta}_*, \boldsymbol{W})p(\boldsymbol{W}|\boldsymbol{D})d\boldsymbol{W}, \tag{5.6}$$

where $r_*$ is prediction of the objective at query point $\boldsymbol{\theta}_*$, $\boldsymbol{D}$ is the training data set consisting of parameters $\boldsymbol{\theta}^{[k]}$ and their objective evaluations $R^{[k]}$. The posterior for $\boldsymbol{W}$ is given by Bayes rule, i.e., $p(\boldsymbol{W}|\boldsymbol{D}) = p(\boldsymbol{D}|\boldsymbol{W})p(\boldsymbol{W})/p(\boldsymbol{D})$. The likelihood function $p(\boldsymbol{D}|\boldsymbol{W})$ is given by

$$p(\boldsymbol{D}|\boldsymbol{W}) = \int p(\boldsymbol{D}|\boldsymbol{W}, \boldsymbol{\beta})p(\boldsymbol{\beta})d\boldsymbol{\beta}, \tag{5.7}$$

[4] $\boldsymbol{W}_{(p\times d)}$ is a projection matrix that projects a vector from a $d$ dimension manifold to a $p$ dimension manifold.




(a) Rosenbrock (b) Rastrigin (c) Noisy Function

FIGURE 5.1: Comparison of stochastic search methods for optimizing the uni-modal Rosenbrock (a) and the multi modal (b) Rastrigin function. (c) Comparison for a noisy objective function. All results show that MORE clearly outperforms other methods.

where $p(\boldsymbol{D}|\boldsymbol{W},\boldsymbol{\beta})$ is the likelihood of the linear model $\beta$ and $p(\boldsymbol{\beta})$ its prior. For the likelihood of the linear model we use a multiplicative noise model, i.e., the higher the absolute value of the objective, the higher the variance. The intuition behind this choice is that we are mainly interested in minimizing the relative error instead of the absolute error[5]. Our likelihood and prior is therefore given by

$$p(\boldsymbol{D}|\boldsymbol{W},\boldsymbol{\beta}) = \prod_{k=1}^{N} \mathcal{N}(R^{[k]}|\boldsymbol{\phi}(\boldsymbol{W}\boldsymbol{\theta}^{[k]})\boldsymbol{\beta}, \sigma^2|R^{[k]}|), \quad p(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \tau^2\boldsymbol{I}), \quad (5.8)$$

Equation 5.7 is a weighted Bayesian linear regression model in $\boldsymbol{\beta}$ where the weight of each sample is scaled by the absolute value of $|R^{[k]}|^{-1}$. Therefore, $p(\boldsymbol{D}|\boldsymbol{W})$ can be obtained efficiently in closed form. However, due to the feature transformation, the output $R^{[k]}$ depends non-linearly on the projection $\boldsymbol{W}$. Therefore, the posterior $p(\boldsymbol{W}|\boldsymbol{D})$ cannot be obtained in closed form any more. We use a simple sample-based approach in order to approximate the posterior $p(\boldsymbol{W}|\boldsymbol{D})$. We use $K$ samples from the prior $p(\boldsymbol{W})$ to approximate the integrals in Equation (5.6) and in $p(\boldsymbol{D})$. In this case, the predictive model is given by

$$p(r_*|\boldsymbol{\theta}_*, \boldsymbol{D}) \approx \frac{1}{K}\sum_i p(r_*|\boldsymbol{\theta}_*, \boldsymbol{W}_i)\frac{p(\boldsymbol{D}|\boldsymbol{W}_i)}{p(\boldsymbol{D})}, \quad (5.9)$$

where $p(\boldsymbol{D}) \approx 1/K \sum_i p(\boldsymbol{D}|\boldsymbol{W}_i)$. The prediction for a single $\boldsymbol{W}_i$ can again be obtained by a standard Bayesian linear regression. Our algorithm is only interested

[5] We observed empirically that such relative error performs better if we have non-smooth objective functions with a large difference in the objective values. For example, an error of 10 has a huge influence for an objective value of $-1$, while for a value of $-10000$, such an error is negligible.

in the expectation $\mathcal{R}_{\boldsymbol{\theta}} = \mathbb{E}[r|\boldsymbol{\theta}]$ in the form of a quadratic model. Given a certain $\boldsymbol{W}_i$, we can obtain a single quadratic model from $\boldsymbol{\phi}(\boldsymbol{W}_i\boldsymbol{\theta})\boldsymbol{\mu}_\beta$, where $\boldsymbol{\mu}_\beta$ is the mean of the posterior distribution $p(\boldsymbol{\beta}|\boldsymbol{W}, \boldsymbol{D})$ obtained by Bayesian linear regression. The expected quadratic model is then obtained by a weighted average over all $K$ quadratic models with weight $p(\boldsymbol{D}|\boldsymbol{W}_i)/p(\boldsymbol{D})$. Note that with a higher number of projection matrix samples(K), the better the posterior can be approximated. Generating these samples is typically inexpensive as it just requires computation time but no evaluation of the objective function. We also investigated using more sophisticated sampling techniques such as elliptical slice sampling [69] which achieved a similar performance but considerably increased computation time. Further optimization of the sampling technique is part of future work.

## 5.4 Experiments

We compare MORE with state of the art methods in stochastic search and policy search such as CMA-ES [32], NES [92], PoWER [53] and episodic REPS [55]. In our first experiments, we use standard optimization test functions [67], such as the the Rosenbrock (uni modal) and the Rastrigin (multi modal) functions. We use a 15 dimensional version of these functions.

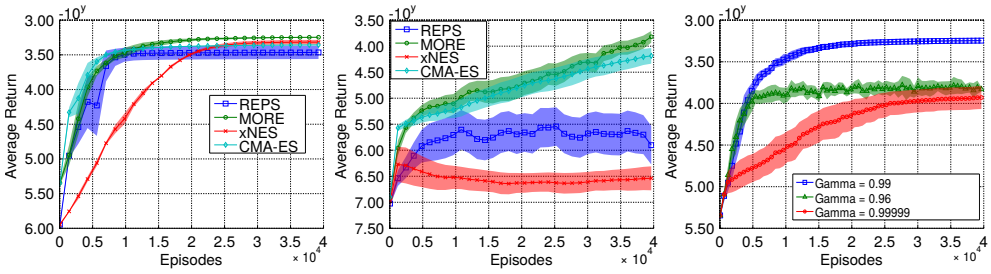
Furthermore, we use a 5-link planar robot that has to reach a given point in task space as a toy task for the comparisons. The resulting policy has 25 parameters, but we also test the algorithms in high-dimensional parameter spaces by scaling the robot up to 30 links (150 parameters). We subsequently made the task more difficult by introducing hard obstacles, which results in a discontinuous objective function. We denote this task hole-reaching task. Finally, we evaluate our algorithm on a physical simulation of a robot playing beer pong. The used parameters of the algorithms and a detailed evaluation of the parameters of MORE can be found in the supplement.

### 5.4.1 Standard Optimization Test Functions

We chose one uni-modal functions $f(\boldsymbol{x}) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$, also known as Rosenbrock function and a multi-modal function which is known as the Rastgirin function $f(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)]$. All these functions have a global minimum equal $f(\boldsymbol{x}) = 0$. In our experiments, the mean of the initial distributions has been chosen randomly.

**Algorithmic Comparison.** We compared our algorithm against CMA-ES, NES, PoWER and REPS. In each iteration, we generated 15 new samples [6]. For MORE,

[6] We use the heuristics introduced in [32, 92] for CMA-ES and NES




(a) Reaching Task (b) High-D Reaching Task (c) Evaluation of $\gamma$

FIGURE 5.2: (a) Algorithmic comparison for a planar task (5 joints, 25 parameters). MORE outperforms all the other methods considerably.(b) Algorithmic comparison for a high-dimensional task (30 joints, 150 parameters). The performance of NES degraded while MORE could still outperform CMA-ES. (c) Evaluation of the entropy bound $\gamma$. For a low $\gamma$, the entropy bound is not active and the algorithm converges prematurely. If $\gamma$ is close to one, the entropy is reduced too slowly and convergence takes long.

REPS and PoWER, we always keep the last $L = 150$ samples, while for NES and CMA-ES only the 15 current samples are kept[7]. As we can see in the Figure 7.3, MORE outperforms all the other methods in terms of learning speed and final performance in all test functions. However, in terms of the computation time, MORE was 5 times slower than the other algorithms. Yet, MORE was sufficiently fast as one policy update took less than $1s$.

**Performance on a Noisy Function.** We also conducted an experiment on optimizing the Sphere function where we add multiplicative noise to the reward samples, i.e., $y = f(\boldsymbol{x}) + \epsilon|f(\boldsymbol{x})|$, where $\epsilon \sim \mathcal{N}(0, 1.0)$ and $f(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{M}\boldsymbol{x}$ with a randomly chosen $\boldsymbol{M}$ matrix.

Figure 7.3(c) shows that MORE successfully smooths out the noise and converges, while other methods diverge. The result shows that MORE can learn highly noisy reward functions.

### 5.4.2 Planar Reaching and Hole Reaching

We used a 5-link planar robot with DMPs [41] as the underlying control policy. Each link had a length of $1m$. The robot is modeled as a decoupled linear dynamical system. The end-effector of the robot has to reach a via-point $\boldsymbol{v}_{50} = [1, 1]$ at time step 50 and at the final time step $T = 100$ the point $\boldsymbol{v}_{100} = [5, 0]$ with its end effector. The reward was given by a quadratic cost term for the two via-points as

[7] NES and CMA-ES algorithms typically only use the new samples and discard the old samples. We also tried keeping old samples or getting more new samples which decreased the performance considerably.

well as quadratic costs for high accelerations. Note that this objective function is highly non-quadratic in the parameters as the via-points are defined in end effector space. We used $5$ basis functions per degree of freedom for the DMPs while the goal attractor for reaching the final state was assumed to be known. Hence, our parameter vector had $25$ dimensions. The setup, including the learned policy is shown in the supplement.

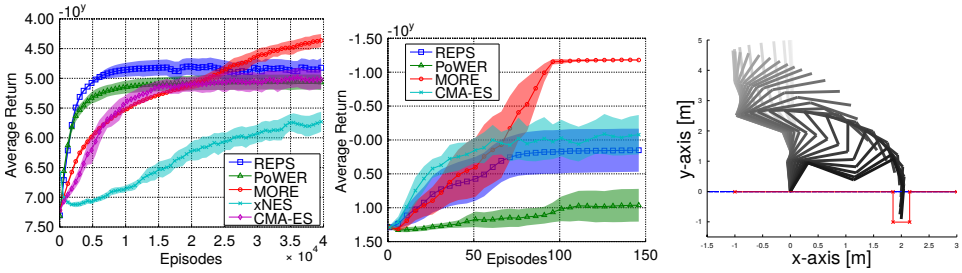
**Algorithmic Comparison.** We generated 40 new samples. For MORE, REPS, we always keep the last $L = 200$ samples, while for NES and CMA-ES only the $40$ current samples are kept. We empirically optimized the open parameters of the algorithms by manually testing 50 parameter sets for each algorithm. The results shown in Figure 5.2(a) clearly show that MORE outperforms all other methods in terms of speed and the final performance.

**Entropy Bound.** We also evaluated the entropy bound in Figure 5.2(c). We can see that the entropy constraint is a crucial component of the algorithm to avoid the premature convergence.

**High-Dimensional Parameter Spaces.** We also evaluated the same task with a 30-link planar robot, resulting in a 150 dimensional parameter space. We compared MORE, CMA, REPS and NES. While NES considerably degraded in performance, CMA and MORE performed well, where MORE found considerably better policies (average reward of -6571 versus -15460 of CMA-ES), see Figure 5.2(b). The setup with the learned policy from MORE is depicted in the supplement.

We use the same robot setup as in the planar reaching task for hole reaching task. For completing the hole reaching task, the robot's end effector has to reach the bottom of a hole ($35$cm wide and $1$ m deep) centering at $[2, 0]$ without any collision with the ground or the walls, see Figure 5.3(c). The reward was given by a quadratic cost term for the desired final point, quadratic costs for high accelerations and additional punishment for collisions with the walls. Note that this objective function is discontinuous due to the costs for collisions. The goal attractor of the DMP for reaching the final state in this task is unknown and is also learned. Hence, our parameter vector had $30$ dimensions.

**Algorithmic Comparison.** We used the same learning parameters as for the planar reaching task. The results shown in Figure 5.3(a) show that MORE clearly outperforms all other methods. In this task, NES could not find any reasonable solution while Power, REPS and CMA-ES could only learn sub-optimal solutions. MORE could also achieve the same learning speed as REPS and CMA-ES, but would then also converge to a sub-optimal solution.

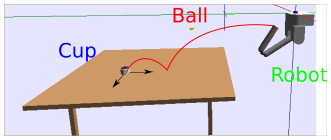



(a) Hole Reaching Task (b) Beer Pong Task (c) Hole Reaching Task Posture

FIGURE 5.3: (a) Algorithmic comparison for the hole reaching task. MORE could find policies of much higher quality. (b) Algorithmic comparison for the beer pong task. Only MORE could reliably learn high-quality policies while for the other methods, even if some trials found good solutions, other trials got stuck prematurely.

### 5.4.3 Beer Pong

In this task, a seven DoF simulated barrett WaM robot arm had to play beer-pong, i.e., it had to throw a ball such that it bounces once on the table and falls into a cup. The ball was placed in a container mounted on the end-effector. The ball could leave the container by a strong deceleration of the robot's end-effector. We again used a DMP as underlying policy representation, where we used the shape parameters (five per DoF) and the goal attractor (one per DoF) as parameters. The mean of our search distribution was initialized with imitation learning. The cup was placed at a distance of 2.2m from the robot and it had a height of 7cm. As reward function, we computed the point of the ball trajectory after the bounce on the table, where the ball is passing the plane of the entry of the cup. The reward was set to be 20 times the negative squared distance of that point to the center of the cup while punishing the acceleration of the joints. We evaluated MORE, CMA, PoWER and REPS on this task. The setup is shown in Figure 5.4 and the learning curve is shown in Figure 5.3(b). MORE was able to accurately hit the ball into the cup while the other algorithms couldn't find a robust policy.




(a) Beer Pong Task

FIGURE 5.4: The Beer Pong Task. The robot has to throw a ball such that it bounces of the table and ends up in the cup.

## 5.5 Conclusion

Using KL-bounds to limit the update of the search distribution is a wide-spread idea in the stochastic search community but typically requires approximations. In this

chapter, we presented a new model-based stochastic search algorithm that computes the KL-bound analytically. By relying on a Gaussian search distribution and on locally learned quadratic models of the objective function, we can obtain a closed form of the information theoretic policy update. We also introduced an additional entropy term in the formulation that is needed to avoid premature shrinkage of the variance of the search distribution. Our algorithm considerably outperforms competing methods in all the considered scenarios. The main disadvantage of MORE is the number of parameters. However based on our experiments, these parameters are not problem specific.

# Chapter 6

# Non-Parametric Contextual Stochastic Search

Stochastic search algorithms are black-box optimizer of an objective function. They have recently gained a lot of attention in operations research, machine learning and policy search of robot motor skills due to their ease of use and their generality. Yet, many stochastic search algorithms require relearning if the task or objective function changes slightly to adapt the solution to the new situation or the new context. In this chapter, we consider the contextual stochastic search setup. Here, we want to find multiple good parameter vectors for multiple related tasks, where each task is described by a continuous context vector. Hence, the objective function might change slightly for each parameter vector evaluation of a task or context. Contextual algorithms have been investigated in the field of policy search, however, the search distribution typically uses a parametric model that is linear in the some hand-defined context features. Finding good context features is a challenging task, and hence, non-parametric methods are often preferred over their parametric counter-parts. In this research, we propose a non-parametric contextual stochastic search algorithm that can learn a non-parametric search distribution for multiple tasks simultaneously. In difference to existing methods, our method can also learn a context dependent covariance matrix that guides the exploration of the search process. We illustrate its performance on several non-linear contextual tasks.

## 6.1 Introduction

Stochastic search algorithms are gradient-free black-box optimizers of some objective function dependent on a high dimensional parameter vector. These algorithms only make weak assumption on the structure of underlying objective function. They only use the objective function values of the parameters that we want to optimise and don't require gradients or higher derivatives of the objective function. For example, in robotics, we can directly evaluate the objective function value for a parameter

vector of a controller by executing that parameter vector and using the return of an episode. Stochastic search algorithms [32, 92, 83] typically maintain a search distribution over the parameters that we want to optimise. This search distribution is used to create samples of the parameter vector. Subsequently, the performance of the sampled parameters is evaluated. Using the samples and their evaluations, a new search distribution is computed by either computing gradient based updates [92, 83], evolutionary strategies [32], the cross-entropy method [61], path integrals [94], or information-theoretic policy updates [55, 4]. However, many of the mentioned algorithms can not be applied for multi-task learning. Therefore, if the task setup or objective function changes slightly, relearning is needed to adapt the solution to the new situation or the new context. For example, consider optimising the parameters of a humanoid soccer robot controller to kick a ball. Once the characteristics of the ball, such as weight of ball, or objective function, such as the desired kick distance changes, relearning of the desired kicking motion is needed. Therefore we would like to learn a context-dependent function that generates optimal parameters for a desired task or context. Contextual search algorithms such as contextual REPS [55] have been investigated in the field of policy search. These algorithms maintain a parametric context-dependent function as the mean of a Gaussian policy which is linear in the some hand-defined context features. Firstly finding good context features to capture the non-linearity of the desired context-dependent function is a challenging task, and hence, non-parametric methods are often preferred over their parametric counter-parts. Secondly only the mean of the Gaussian search distribution is context-dependent while the covariance matrix is fixed for all contexts. Hence, these algorithms find a covariance matrix that, in average, is good for all the contexts. However, in order to guide the policy search it is desirable to have a fully context-dependent search distribution with optimal mean and covariance matrix for a specific context. Therefore, we introduce a non-parametric contextual policy search method that can learn non-linear context-dependent functions and leverage from a fully context-dependent search distribution. We name our method local Covariance Estimation with Controlled Entropy Reduction (local CECER). We will show that local CECER performs favourably.

### 6.1.1 Problem Statement

Given a query context vector $\boldsymbol{s}^*$ with $m$ dimensions which defines a task, we want to find a non parametric context-dependent function $m_*(\boldsymbol{s}) : \mathbb{R}^m \rightarrow \mathbb{R}^n$ that generates a parameter vector $\boldsymbol{\theta}^*$ with $n$ dimensions such that it maximizes an objective function $R(\boldsymbol{\theta}, \boldsymbol{s}) : \{\mathbb{R}^n, \mathbb{R}^m\} \rightarrow \mathbb{R}$. The only accessible information on $R(\boldsymbol{\theta}, \boldsymbol{s})$ are evaluations $\{R^{[k]}\}_{k=1...N}$ of samples $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}\}_{k=1...N}$, where $k$ is the index of the sample

and $N$ is number of samples. Essentially the goal of local CECER is to generate a dataset $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}\}_{k=1...N}$ that contains the optimal parameters for the corresponding context vectors. With this data set, the optimal vector $\boldsymbol{\theta}^*$ for a given context $\boldsymbol{s}^*$ can be found in a non-parametric fashion using locally weighted linear regression method.

### 6.1.2 Related Work

In order to generalize a parameter vector to the other contexts, for example, kicking the ball for different distances, typically the parameters are optimized for several target contexts independently. Subsequently, regression methods are used to generalize the optimized contexts to a new, unseen context. Although such approaches have been used successfully, they are time consuming and inefficient in terms of the number of needed training samples as optimizing for different contexts and the generalization between optimized parameters for different contexts are two independent processes[16]. Hence, it is desirable to learn the selection of the parameter for multiple tasks at once without restarting the learning process once we see a new task. This problem setup is also known as contextual policy search [55, 51]. Such a multi-task learning capability was established for information-theoretic policy search algorithms [75], such as the Contextual Relative Entropy Policy Search (CREPS) algorithm [55]. Contextual REPS was originally applicable for the problems with linear generalization over contexts or tasks. In [2], contextual REPS was extended for tasks with non-linear generalization over contexts, by using radial basis functions resulting in contextual RBF-REPS. However due to use of radial basis functions, this method can suffer from curse of dimensionality and also finding a good settings for RBFs is a challenging task. Moreover, the update rule of the search distribution in REPS and RBF-REPS is not fully context-dependent. In addition, REPS and RBF-REPS can suffer from premature convergence. In [5] Covariance Estimation with Controlled Entropy Reduction (CECER) algorithm was introduced to alleviate the premature convergence problem of REPS. Our new algorithm local CECER leverage from both nonlinear generalization over contexts and fully context dependent search distribution update rule while it also uses CECER algorithm concept[5] to avoid premature convergence.

## 6.2 Non-Parametric Contextual Stochastic Search

local CECER is a non-parametric policy search approach. Therefore we always maintain a dataset $\mathcal{D} = \{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}, \boldsymbol{\Sigma}^{[k]}\}_{k=1...N}$ with $N$ samples that contains the contexts, parameters pair $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}\}$ and its evaluation $R^{[k]}$ as well as the covariance

**Algorithm 4** local CECER Weights Computation

**Input :** Data Set $\mathcal{D}\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}, R^{[k]}, \boldsymbol{\Sigma}^{[k]}\}_{k=1...N}$, the query context $\boldsymbol{s}^*$
**Compute the locality weightings $w^{[k]}$ for each sample:**

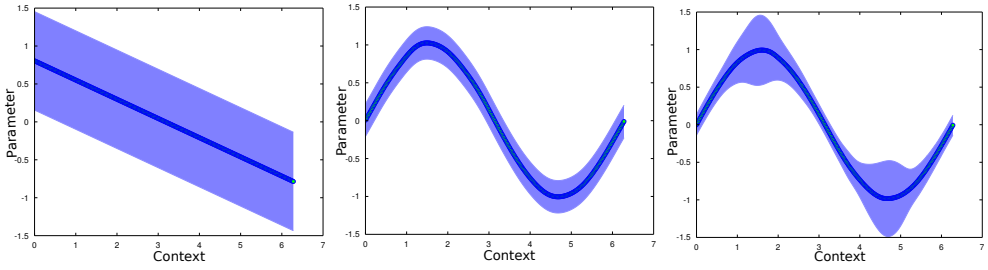
$$w^{[k]} = \exp(-0.5|\boldsymbol{s}^{[k]} - \boldsymbol{s}^*|^2/b) \, , Z_w = \sum_{k=1}^{N} w^{[k]}.$$

**Compute the weights $d^{[k]}$ for each sample:**
1- Optimize the dual function $g$ for $\eta$ and $\boldsymbol{w}$

$$\begin{aligned} g(\eta, \boldsymbol{w}) =& \eta\epsilon + \hat{\boldsymbol{\phi}}^T \boldsymbol{w} \\ &+ \eta \log \left( \sum_{K=1}^{N} \frac{w^{[k]}}{Z_w} \exp \left( \frac{R^{[k]} - \boldsymbol{\phi}(\boldsymbol{s}^{[k]})^T \boldsymbol{w}}{\eta} \right) \right) \\ \hat{\boldsymbol{\phi}} =& \sum_{k=1}^{N} \frac{w^{[k]}}{Z_w} \boldsymbol{\phi}(\boldsymbol{s}^{[k]}). \end{aligned}$$

2- Compute weights

$$d^{[k]} = \frac{w^{[k]} \exp \left( \frac{R^{[k]} - \boldsymbol{\phi}(\boldsymbol{s}^{[k]})^T \boldsymbol{w}}{\eta} \right)}{Z} \, , \; Z = \sum_{k=1}^{N} d^{[k]}.$$




(a) Contextual CECER (b) Contextual RBF-CECER (c) Local CECER

FIGURE 6.1: The learned policy by CECER, RBF-CECER and local CECER for sin task. Darker blue shows the mean of the search distribution for each context. While the shaded area with lighter blue shows the variance of the search distribution around the mean for each context. The results show that Local CECER and RBf-CECER can learn non-linear policies. Moreover local CECER is able to learn a search distribution that both the mean and variance of the distribution is context dependent which is a desirable feature. As you can see in CECER and RBF-CECER the variance of the search distribution for all the contexts is fixed.

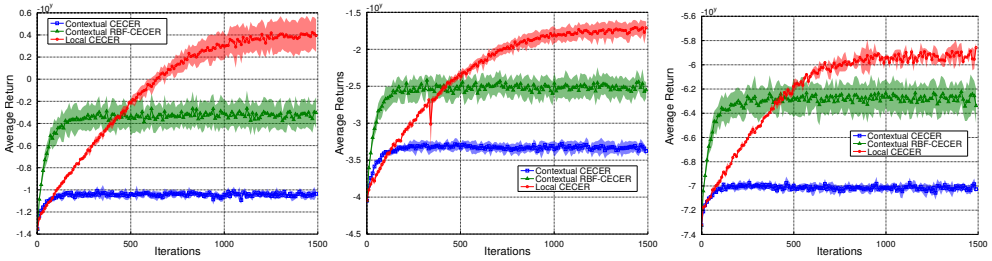
matrix $\boldsymbol{\Sigma}^{[k]}$ that has been used to generate parameters $\boldsymbol{\theta}^{[k]}$. In each iteration, given a new query context $\boldsymbol{s}^*$, we first compute a locality (similarity) weighting $w^{[k]}$ for each sample with respect to the query context $\boldsymbol{s}^*$. We use these locality weightings to compute a weight or pseudo probability $d^{[k]}$ for each sample in the data set and subsequently, we obtain a local Gaussian search distribution $\pi_*(\boldsymbol{\theta}|\boldsymbol{s})$. We use the search distribution $\pi_*(\boldsymbol{\theta}|\boldsymbol{s}^*)$ to create a sample $\boldsymbol{\theta}^*$ for the query context $\boldsymbol{s}^*$. Subsequently, the evaluation $R^*$ of $\{\boldsymbol{s}^*, \boldsymbol{\theta}^*\}$ is obtained by querying the objective function $R(\boldsymbol{\theta}^*, \boldsymbol{s}^*)$. Afterwards, we update the dataset with the new sample $\{\boldsymbol{s}^*, \boldsymbol{\theta}^*, \boldsymbol{\Sigma}^*, R^*\}$ [1]. We also use the locality weightings to update the covariance matrices of neighboured samples of query context $\boldsymbol{s}^*$ to improve the estimate of their local covariance matrix. This process will run iteratively until a stopping criteria is met. We start by explaining how the weights or pseudo probabilities $d^{[k]}$ are computed and, after that, we explain the local Gaussian search distribution update rules.

### 6.2.1 Weight Computation

Given query context $\boldsymbol{s}^*$, local CECER first computes a locality weighting $w^{[k]}$ for each sample. We use a normalized squared exponential kernel i.e.,

$$w(\boldsymbol{s}) = \frac{k(\boldsymbol{s}, \boldsymbol{s}^*)}{\int k(\boldsymbol{s}, \boldsymbol{s}^*) d\boldsymbol{s}} \ , \ k(s, s^*) = \exp(-0.5|\boldsymbol{s} - \boldsymbol{s}^*|^2/b).$$

[1] Please note that the way we sample contexts $\boldsymbol{s}^{[k]}$ depends on the task. Throughout this chapter we use a uniform distribution to sample contexts $\boldsymbol{s}$.




(a) Sphere (b) Rosenbrock (c) Cigar

FIGURE 6.2: The performance comparison of stochastic search methods for optimising contextual version of standard functions (a)Sphere, (b)Rosenbrock and (c)Cigar, The results show that local CECER outperforms both CECER and RBF-CECER.

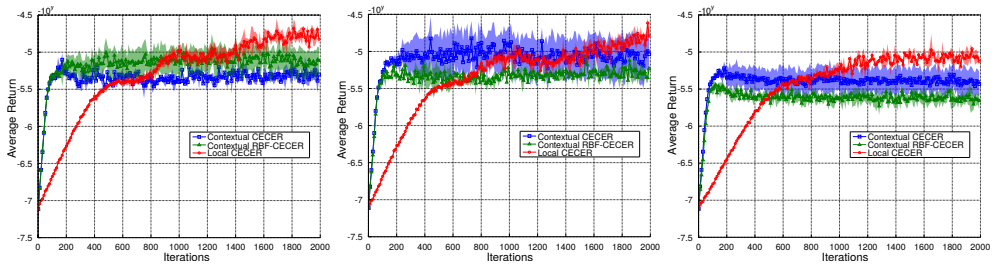
Now we use this locality weightings to obtain a pseudo probability or a weight for each sample in our data set. To do so we find the joint probabilities $p_*(\boldsymbol{s}, \boldsymbol{\theta}) = \pi_*(\boldsymbol{\theta}|\boldsymbol{s})\mu_*(\boldsymbol{s})$ by optimizing the following performance criteria[55] for each new query context $\boldsymbol{s}^*$, i.e.,

$$\begin{aligned} &\max_{p_*} \iint p_*(\boldsymbol{s}, \boldsymbol{\theta}) \mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}} d\boldsymbol{s} d\boldsymbol{\theta} \\ &\text{s.t. } \epsilon \geq \text{KL}(p_*(\boldsymbol{s}, \boldsymbol{\theta}) || \mu_*(\boldsymbol{s}) q(\boldsymbol{\theta}|\boldsymbol{s})), \\ &\hat{\boldsymbol{\phi}} = \iint p_*(\boldsymbol{s}, \boldsymbol{\theta}) \boldsymbol{\phi}(\boldsymbol{s}) d\boldsymbol{s} d\boldsymbol{\theta}, \\ &1 = \iint p_*(\boldsymbol{s}, \boldsymbol{\theta}) d\boldsymbol{s} d\boldsymbol{\theta}. \end{aligned} \tag{6.1}$$

The key idea behind this optimization program is to ensure a smooth and stable learning process by bounding the Kullback-Leibler divergence between the old local search distribution and the newly estimated local search distribution while maximising the expected return for the given context $\boldsymbol{s}^*$. Where $\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}}$ denotes the expected performance when evaluating parameter vector $\boldsymbol{\theta}$ in context $\boldsymbol{s}$, $q$ is the old sample distribution. While $\mu_(\boldsymbol{s})$ is the context distribution, $\mu_*(\boldsymbol{s})$ denotes the local context distribution with respect to context $\boldsymbol{s}^*$ which can be obtained using the locality weighting function $w(\boldsymbol{s})$ i.e.,

$$\mu_*(\boldsymbol{s}) = \frac{w(\boldsymbol{s})\mu(s)}{\int \mu(\boldsymbol{s}) w(\boldsymbol{s}) d\boldsymbol{s}}.$$

In addition $\hat{\boldsymbol{\phi}} = \int_{\boldsymbol{s}} \mu_*(\boldsymbol{s}) \boldsymbol{\phi}(\boldsymbol{s}) d\boldsymbol{s}$ is the expected feature vector for the local context distribution $\mu_*(\boldsymbol{s})$, a given query context $\boldsymbol{s}^*$ and a given feature space $\boldsymbol{\phi}$. This optimization problem can be solved efficiently by the method of Lagrangian multipliers




(a) 1 dim context (b) 2 dim contexts (c) 3 dim contexts

FIGURE 6.3: Performance evaluation on hole reaching task up to 3 dim contextual setup. The results show that local CECER outperforms other algorithms and can learn the task while the other algorithms can not learn the task. Please also see figure 6.4 and figure 6.5

[12]. The solution for $p_*(\boldsymbol{s}, \boldsymbol{\theta})$ is now given by

$$p_*(\boldsymbol{s}, \boldsymbol{\theta}) \propto q(\boldsymbol{\theta}|\boldsymbol{s}) \mu_*(\boldsymbol{s}) \exp\left((\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}} - V(\boldsymbol{s}))/\eta\right),$$

where $V(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^T \boldsymbol{w}$ is a context dependent baseline which is subtracted from the return $\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}}$. The parameters $\boldsymbol{w}$ and $\eta$ are Lagrangian multipliers that can be obtained by optimizing the dual function, given as

$$\begin{aligned} g(\eta, \boldsymbol{w}) =& \eta\epsilon + \hat{\boldsymbol{\phi}}^T \boldsymbol{w} \\ &+ \eta \log \left( \iint \mu_*(\boldsymbol{s}) q(\boldsymbol{\theta}|\boldsymbol{s}) \exp\left( \frac{\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}} - \boldsymbol{\phi}(\boldsymbol{s})^T \boldsymbol{w}}{\eta} \right) d\boldsymbol{\theta} d\boldsymbol{s} \right). \end{aligned} \tag{6.2}$$

This policy update results in a weight or pseudo probability

$$d^{[k]} = w^{[k]} \exp\left((R^{[k]} - V(\boldsymbol{s}^{[k]}))/\eta\right)$$

for each sample $[\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}]$ given a query context $\boldsymbol{s}^*$ where

$$w^{[k]} = \frac{k(\boldsymbol{s}, \boldsymbol{s}^*)}{Z_w}, Z_w = \sum_{k=1}^{N} w^{[k]}.$$

See Algorithm 1 for a compact representation of the weight computation of local CECER algorithm. In the next section we show how we can use these pseudo probabilities to estimate a local Gaussian search distribution $\pi_*(\boldsymbol{\theta}|\boldsymbol{s})$ exclusively for the query context $\boldsymbol{s}^*$.

### 6.2.2 Search Distribution Update Rule

Given dataset $\{\boldsymbol{s}^{[k]}, \boldsymbol{\theta}^{[k]}, w^{[k]}, \boldsymbol{\Sigma}^{[k]}, d^{[k]}\}_{k=1...N}$ and a query context $\boldsymbol{s}^*$, we want to find a local linear Gaussian search distribution

$$\pi_*(\boldsymbol{\theta}|\boldsymbol{s}) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{m}_{\pi_*}(\boldsymbol{s}) = \boldsymbol{A}_{\pi_*}^T\varphi(\boldsymbol{s}), \boldsymbol{\Sigma}_{\pi_*}\right),$$

by finding $\boldsymbol{A}_{\pi_*}$ and $\boldsymbol{\Sigma}_{\pi_*}$. Where $\varphi(\boldsymbol{s})$ is an arbitrary feature function of context $\boldsymbol{s}$, $\boldsymbol{A}_{\pi_*}^T$ is the gain matrix and $\boldsymbol{\Sigma}_{\pi_*}$ is the covariance matrix. Throughout this chapter $\boldsymbol{\varphi}(\boldsymbol{s}) = [1 \ \boldsymbol{s}]$, which results in linear generalization over contexts. Therefore we need update rules for updating the mean function $\boldsymbol{m}_{\pi_*}$ and for updating the covariance matrix $\boldsymbol{\Sigma}_{\pi_*}$.

#### Context-Dependent Mean-Function

In order to find $\boldsymbol{m}_{\pi_*}$, the parameters $\boldsymbol{A}_{\pi_*}$ can be obtained by the weighted linear ridge regression

$$\boldsymbol{A}_{\pi_*} = \left(\boldsymbol{\Phi}^T\boldsymbol{D}\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{D}\boldsymbol{U}, \tag{6.3}$$

where $\boldsymbol{\Phi}^T = [\boldsymbol{\varphi}^{[1]}, ..., \boldsymbol{\varphi}^{[N]}]$ contains the feature vector for all samples, $\boldsymbol{U}^T = [\boldsymbol{\theta}^{[1]}, ..., \boldsymbol{\theta}^{[N]}]$ contains all the sample parameters, $\boldsymbol{D}$ is the diagonal weighting matrix containing the weightings $d^{[k]}$ and $\lambda\boldsymbol{I}$ is a regularization term.

#### Context-Dependent Covariance Matrix

Similar to Standard Contextual REPS we can directly use the weighted sample covariance matrix $\boldsymbol{S}_*$ as local covariance estimate $\boldsymbol{\Sigma}_{\pi_*}$ which is obtained by

$$\begin{aligned}\boldsymbol{S}_* &= \frac{\sum_{k=1}^N d^{[k]}\left(\boldsymbol{\theta}^{[k]} - \boldsymbol{A}_{\pi_*}^T\boldsymbol{\varphi}(\boldsymbol{s}^{[k]})\right)\left(\boldsymbol{\theta}^{[k]} - \boldsymbol{A}_{\pi_*}^T\boldsymbol{\varphi}(\boldsymbol{s}^{[k]})\right)^T}{Z}, \\ Z &= \frac{(\sum_{k=1}^N d^{[k]})^2 - \sum_{k=1}^N (d^{[k]})^2}{\sum_{k=1}^N (d^{[k]})}.\end{aligned} \tag{6.4}$$

However it has been shown that the sample covariance matrix from Equation 7.2 can cause premature convergence [5]. In order to alleviate this problem, similar to CECER [5] we combine the local old covariance matrix and the local sample covariance matrix from Equation 7.2, i.e.,

$$\boldsymbol{\Sigma}_{\pi_*} = (\lambda)\boldsymbol{S}_* + (1-\lambda)\boldsymbol{\Sigma}_{q_*}.$$

In local CECER, the local old covariance matrix also depends on context query $\boldsymbol{s}^*$. Therefore we estimate the local old covariance $\boldsymbol{\Sigma}_{q_*}$ by a weighted average of covariance matrices $\boldsymbol{\Sigma}^{[k]}$ in the dataset. We use the locality weightings $w^{[k]}$ as weights, i.e.,

$$\boldsymbol{\Sigma}_{q_*} = \sum_{k=1}^{N} \frac{w^{[k]}}{Z_w} \boldsymbol{\Sigma}^{[k]}.$$

There are different ways to determine the interpolation factor $\lambda \in [0, 1]$ between the sample covariance matrix $\boldsymbol{S}_*$ and the old covariance matrix $\boldsymbol{\Sigma}_{q_*}$. For example, see the rank-$\mu$ update in CMA-ES algorithm [32]. Similar to CECER, the factor $\lambda \in [0, 1]$ is chosen in such a way that the entropy of the new search distribution is reduced by a certain amount $\Delta H$. The entropy of a Gaussian distribution only depends on its covariance $\boldsymbol{\Sigma}_{\pi_*}$ and is given by

$$H(\boldsymbol{\Sigma}_{\pi_*}) = 0.5\big(n + n\log(2\pi) + \log|\boldsymbol{\Sigma}_{\pi_*}|\big).$$

Therefore, $\lambda$ is chosen such that a desired entropy reduction is achieved, i.e.,

$$H(\boldsymbol{\Sigma}_{q_*}) - H(\lambda\boldsymbol{\Sigma}_{q_*} + (1-\lambda)\boldsymbol{S}_*) = \Delta H.$$

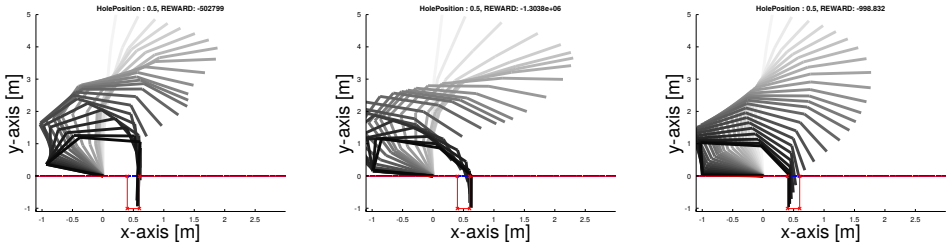
The parameter $\Delta H$ is a user-defined parameter to tune the algorithm. After obtaining $\boldsymbol{\Sigma}_{\pi_*}$ we update all the covariance matrices in the dataset using locality weightings i.e.,

$$\boldsymbol{\Sigma}^{[k]} = \beta\boldsymbol{\Sigma}_{\pi_*} + (1-\beta)\boldsymbol{\Sigma}^{[k]}, \beta = \frac{w^{[k]}}{Z_w}.$$

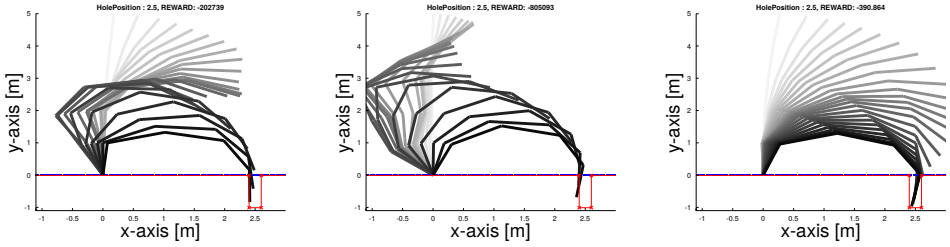
We subsequently use the policy $\pi_*(\boldsymbol{\theta}|\boldsymbol{s}^*)$ to generate a new parameter $\boldsymbol{\theta}^*$ for the context query $\boldsymbol{s}^*$ and add the new sample $\{\boldsymbol{s}^*, \boldsymbol{\theta}^*, R^*, \boldsymbol{\Sigma}_{\pi_*}\}$ to our dataset. In this research given that we always want to keep $N$ samples in our dataset, we replace the new sample with the oldest sample if number of samples exceeds $N$. However other dataset update strategies based on context density could be implemented.

## 6.3 Experiments

In this section we compare our algorithm local CECER with contextual CECER and contextual RBF-CECER [5] which are improved versions of standard contextual REPS and RBF-REPS[2] respectively. Contextual RBF-CECER is similar to contextual CECER with the difference that RBF-CECER use radial basis functions for non-linear generalization over contexts[2]. We chose three different contextual toy tasks. We use a simple standard $\sin$ function with one parameter to show that

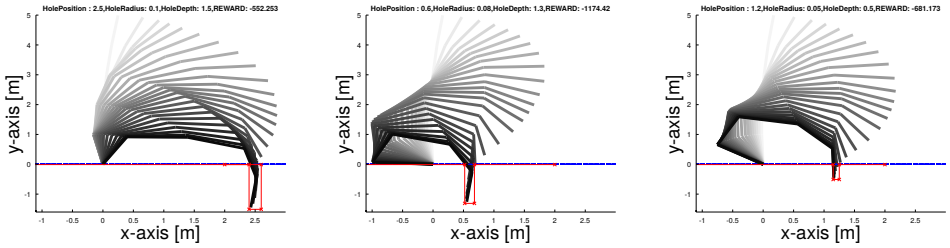



(a) Contextual CECER (b) Contextual RBF-CECER (c) Local CECER




(d) Contextual CECER (e) Contextual RBF-CECER (f) Local CECER

FIGURE 6.4: A 5-link robot has to reach the bottom of a hole (20 cm wide and 1 m deep) at time step 100 centering at a point varying from 0.5 to 2.5 without any collision with the ground or the hole wall. The red lines show the ground and the hole. The postures of the resulting motion are shown as overlay, where darker postures indicate a posture which is close in time to the bottom of the hole. In the title of each figure, you can see the given context value and gained reward by each algorithm. In this task while local CECER successfully complete the task for both contexts, the other algorithms fail to complete the tasks .(Please see the resulting rewards in the title of the figures)




(a) Local CECER-3 dim contexts (b) Local CECER-3 dim contexts (c) Local CECER-3 dim contexts

FIGURE 6.5: The learned policy by local CECER for 3 dimension contextual hole reaching task. As you can see local CECER could learn the task for 3 dim context while the other algorithms didn't learn a reasonable policy that we could show. You can see the value of query contexts as well as obtained reward in the title of figures.

local CECER can learn non-linear policies with context-dependent covariance matrix. In the second series, we use standard optimization test functions [67], such as the Sphere, the Rosenbrock and the Cigar function. We extend these functions to be

applicable for contextual setting with non-linear generalization over contexts. The task is to find the optimum 15 dimensional parameter vector $\boldsymbol{\theta}$ for a given 2 dimensional context $\boldsymbol{s}$. Furthermore for the comparisons we use a 5-link planar robot that has to reach the bottom of a given hole without collision with the walls of the hole in task space. We used dynamic movement primitives (DMPs) [41] as underlying policy representation with 30 parameters (five basis functions per dimension and 1 goal position per dimension). For this task, we use three contexts which are the position of the hole, width and depth of the hole. We use hole reaching task with one dimensional context(hole position), two dimensional context (hole position and hole width) and three dimensional context. Figure 6.4 shows the setup. We show the average as well as two times the standard deviation of the results over 5 trials for each experiment. Note that the y-axis of all plots is in a logarithmic scale.

### 6.3.1 Sinus Function Task

In this task, the reward function is given as the distance to a sin function and the distance punishment varies for the context variable (i.e. some contexts are harder to achieve) i.e., $\mathcal{R}_{\boldsymbol{s}\boldsymbol{\theta}} = -(\boldsymbol{\theta} - \sin(\boldsymbol{s}))^2 \times (1 + 5\cos(\boldsymbol{s}))^2$. Both, context and parameter to learn, are 1 dimensional. In Figure 6.1, we show the mean and variance of the search distribution for each context. Figure 6.1 shows that RBF-CECER and local CECER both can capture the non-linearity of the function, however only local CECER has different search distribution variance for each context. This experiment shows that only local CECER can learn which context is harder to achieve (less variance) which is easier achieve (high variance).

### 6.3.2 Standard Optimization Test Functions

We chose three standard optimization functions which are the Sphere function $f(\boldsymbol{s}, \theta) = \sum_{i=1}^{p} \boldsymbol{x}_i^2$ and the Rosenbrock function $f(\boldsymbol{s}, \theta) = \sum_{i=1}^{p-1} [100(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i^2)^2 + (1 - \boldsymbol{x}_i)^2]$ and also a function which is known as Cigar function $f(\boldsymbol{s}, \theta) = x_1^2 + 10^6 \sum_{i=2}^{p} \boldsymbol{x}_i^2$. Where $\boldsymbol{x} = \boldsymbol{\theta} + \sin(\boldsymbol{A}\boldsymbol{s})$. The matrix $\boldsymbol{A}$ is a constant matrix that was chosen randomly. In our case, because the context $\boldsymbol{s}$ is 2 dimensional, $\boldsymbol{A}$ is a $n \times 2$ dimensional vector. Now, the optimum $\boldsymbol{\theta}$ for these functions is non-linearly dependent on the given context $\boldsymbol{s}$. The initial search area of $\boldsymbol{\theta}$ for all experiments is restricted to the hypercube $-5 \leq \boldsymbol{\theta}_i \leq 5, i = 1, \ldots, p$ and contexts are samples uniformly from interval $0 \leq \boldsymbol{s}_i \leq 3, i = 1, \ldots, z$ where $z$ is dimension of the context space $\boldsymbol{s}$. In our experiments, the mean of the initial distribution to generate the initial data set have been chosen randomly in the defined search area.

**Algorithmic Comparison** We generate 2500 samples in the first iteration and in each iteration, we generated 1 new samples and we always keep last 2500 samples. The results in figure 6.2 shows that local CECER outperforms both contextual CECER and contextual RBF-CECER.

### 6.3.3 Planar Hole Reaching

In this task, we used a 5-link planar robot with DMPs [41] as the underlying control policy. Each link had a length of $1m$. The robot is modelled as a decoupled linear dynamical system. For completing the hole reaching task, the robot end effector has to reach the bottom of a hole with a width varying from $10$cm to $40$cm, centering at a point varying from $0.5$m to $2.5$m and with a depth varying from $50$cm to $1.5$m without any collision with the ground or the hole wall. The reward was given by a quadratic cost term for the desired final point, quadratic costs for high accelerations and quadratic costs for collisions with the environment. Note that this performance function is discontinuous due to the cost for collisions. The DMPs goal attractor for reaching the final state in this task is unknown and need to also be learned. Hence, our parameter vector had $30$ dimensions. The learning setup is shown in Figure 6.4.

**Algorithmic Comparison** For the planar task we generated 2500 samples in the first iteration and 1 new samples in each iteration. We always keep last 2500 samples. We compare all three algorithms in three different contextual settings up to three dimensional context setting. The results in Figure 6.3 shows that local CECER outperforms the other two algorithms in all three different contextual settings. Figure 6.4 and Figure 6.5 shows the learned policies for 1 dimensional context, which is the hole position, and three dimensional context which are the hole position, the hole width and the hole depth. The results show that local CECER could successfully learn for all the query contexts while the other algorithms failed to learn this task.

## 6.4 Conclusion

Multi task learning is an important feature for a robot learning algorithm as a robot usually needs to quickly adapt to new situations. Therefore, in this chapter, we investigated a non-parametric contextual stochastic search method called local CECER. We showed that local CECER leverages from a fully context dependent policy update and it is able to learn non-linear policies. We showed that local CECER outperforms the other contextual algorithms. For the future work we investigate the methods to set the bandwidth of the kernel function automatically.

# Chapter 7

# Humanoid Kick with Controlled Distance

We investigate the learning of a flexible humanoid robot kick controller, i.e., the controller should be applicable for multiple contexts, such as different kick distances, initial robot position with respect to the ball or both. Current approaches typically tune or optimise the parameters of the biped kick controller for a single context, such as a kick with longest distance or a kick with a specific distance. Hence our research question is that, how can we obtain a flexible kick controller that controls the robot (near) optimally for a continuous range of kick distances? The goal is to find a parametric function that given a desired kick distance, outputs the (near) optimal controller parameters. We achieve the desired flexibility of the controller by applying a contextual policy search method. With such a contextual policy search algorithm, we can generalize the robot kick controller for different distances, where the desired distance is described by a real-valued vector. We will also show that the optimal parameters of the kick controller is a non-linear function of the desired distances and a linear function will fail to properly generalize the kick controller over desired kick distances.

## 7.1 Introduction

Designing optimal controllers for robotic systems is one of the major tasks in the robotics research field. Hence, it is desirable to have a controller that can control the robot for different tasks or contexts in real time, for example a soccer robot should be able to kick the ball for any desired kick distance which can be chosen from a continuous range of kick distances. We define a task as a context. Context is a vector of variables that do not change during a task's execution, but might change from task to task. In this research for example, the context is the distance the ball travels after being kicked and can be chosen by the agent. The kick task is one of the most important skills in the context of robotic soccer[24]. Typically the kick controllers are

only applicable for a discretized number of desired distances. For example three sets of parameters for the kick controller is obtained which are applicable for long, mid and short distance kicks. Such a controller limits the robot to properly pass the ball to its teammates. Controlling the robot to kick the ball (near)optimally for different distances, allows the agents have a lot more control and options regarding their next decision, which could affects the game's outcome. Our goal is to find a parametric function that given a desired kick distance, outputs the (near) optimal controller parameters. In the other word we would like to obtain a policy $\pi(\theta|s)$ that sets the parameters $\theta$ of a robot kick controller given a context $s$ which is the desired kick distance. In order to optimize the robot controller parameters given an objective function, there are many algorithms proposed by the scientific community [32, 92, 90, 83, 61, 94, 55, 4]. However, many of these algorithms usually optimize a parameter set for a single context, such as optimizing a kick for the longest distance or the highest accuracy [23]. In other words, these algorithms fail to generalize the optimized movement for a context to different contexts. In order to generalize the kick motion to, for example, different kicking distances, typically the parameters are optimized for several target contexts independently. Afterwards, to generalize movements to new unseen contexts and to obtain a continuous policy $\pi(\theta|s)$, regression methods are commonly used [96, 73]. Although such approaches have been successfully used, they are time consuming and inefficient regarding the number of needed training samples. In such a method, data-points obtained from optimizing the kick controller for context $s$ cannot be re-used to improve and accelerate the optimisation for context $s'$. This is due to the fact that optimizing the controller parameters and generalizing them are two independent processes and the correlation between different contexts is ignored during the optimisation. Therefore in this research we propose to use contextual relative entropy policy search(CREPS) algorithm which searches for the optimal parameters of the policy $\pi(\theta|s)$ in one run optimisation process a. In the other word in CREPS, optimizing the controller parameters and generalizing them happens simultaneously and therefore the correlation between different contexts can be exploited in order to accelerate the optimisation. CREPS, however, has a major drawback related to its search distribution update. The distribution might collapse prematurely to a point-estimate, resulting in premature convergence. On the other hand, the CMA-ES algorithm [32] which is not a contextual algorithm has shown to be able to avoid premature convergence. Therefore we combine the update rules of CREPS and CMA-ES resulting to the contextual relative entropy policy search with covariance matrix adaptation(CREPS-CMA). We will show that CREPS-CMA avoids premature convergence. Hence we will use CREPS-CMA for optimising the kick controller. We will also show that a non-linear function of desired kick distance

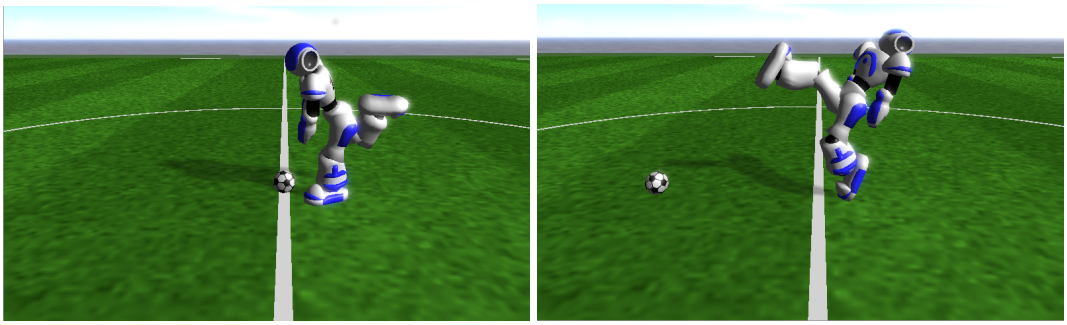


FIGURE 7.1: The initial (left) and final (right) positions of an exemplary kick movement.

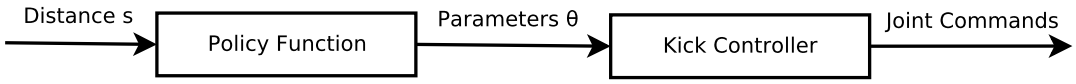



FIGURE 7.2: The pipeline of our contextual kick movement.

clearly outperforms a linear one. This effect has been also observed for the humanoid walking task [1]. Now our robot is able to kick the ball for a continuous range of desire kick distances. This is in contrast with our previous approach where we had 3 sets of parameters for short, mid and long distance kicks.

## 7.2 The Approach

We used a simulated Nao robot shown in Figure 7.1 for our experiments. Our movement pipeline is composed of two main parts: a kick controller, which receives parameters $\boldsymbol{\theta}$ and converts them into joint commands for the robot's servos; and a policy function, which maps a given context $s$ for a specific kick distance into the corresponding parameter vector $\boldsymbol{\theta}$. The pipeline for the kick task, whose context is the kick distance $s$ with a straight kick direction with respect to the torso, is shown in Figure 7.2.

### 7.2.1 Kick Controller

We have a kick controller which is a simple keyframe-based [23] linear model and we also have stability module as in [24] that stabilize the robot during performing the kick movement. A keyframe, as defined in [23], is a complete description of joint angles, either absolute or relative to the previous keyframe. Our keyframe based controller is defined by the following parameters:

- The initial keyframe, represented as a vector $\alpha$ of joint angles with dimension $l$,

- The final keyframe, also represented as a vector $\beta$ of joint angles with dimension $l$.
- The action time $t$ that is the amount of time the robot takes to move from the initial to the final keyframe. The joint angles are linearly interpolated across $t$ to create the corresponding movement.

During performing kick only the legs joints move and remaining joints (arms and head joints) are kept constant. As each leg has 6 joints, $\alpha$ and $\beta$ are 12-dimensional vectors. Therefore considering the action time $t$, our kick controller has 25 parameters to set. The controller receives a 25-dimensional parameter vector $\boldsymbol{\theta}$, which is then interpolated and coded into motor commands. Figure 7.1 shows the initial and final positions of an exemplary kick. The stability module has its constant parameters which doesn't change from task to task, please see [24] for more details of our stability module. Now we need to find a policy function of kick distance $s$ that sets our controller parameters with the proper parameters $\boldsymbol{\theta}$ for any given desired kick distance.

### 7.2.2 Policy Function

Our goal is to find a function in form of

$$\mu(\boldsymbol{s}) = \boldsymbol{A}^T \varphi(\boldsymbol{s}),$$

that given a context vector $\boldsymbol{s}$ with dimension $d_s$, outputs a optimal parameter vector $\boldsymbol{\theta}$ with dimension $d_{\boldsymbol{\theta}}$ such that it maximise our objective function $R(\boldsymbol{\theta}, \boldsymbol{s}) : \{\mathbb{R}^{d_s}, \mathbb{R}^{d_{\boldsymbol{\theta}}}\} \rightarrow \mathbb{R}$. Where $\varphi(\boldsymbol{s})$ is an arbitrary feature function of context $\boldsymbol{s}$ that outputs a feature vector with dimension $d_\varphi$ and the gain matrix $A_\pi$ is a $d_{\boldsymbol{\theta}} \times d_\varphi$ matrix. Typically $\varphi(\boldsymbol{s}^{[i]}) = [1 \;\; \boldsymbol{s}^{[i]}]$, which results in linear generalization over contexts. In order to achieve non-linear generalization over contexts, we can use normalized radial basis features (RBF) as a feature function:

$$\varphi(\boldsymbol{s}^{[i]}) = \frac{\psi_j(\boldsymbol{s}^{[i]})}{\sum_{j=1}^{K} \psi_j(\boldsymbol{s}^{[i]})}, \quad \psi_j(\boldsymbol{s}^{[i]}) = \exp(-\frac{(\boldsymbol{s}^{[i]} - c_j)^2}{2\sigma^2}),$$

where $K$ is the number of RBFs and centres $\{c_j\}_{j=1\ldots K}$ are equally spaced in the range of $\boldsymbol{s}$, based on the desired number of RBFs $K$, and $\sigma^2$ is the bandwidth of the RBF. The bandwidth represents how related contexts are. A large bandwidth means that contexts are very similar and therefore the relationship is (near)linear. A bandwidth of 0 is an extreme case where movements are not generalizable at all, and each context has its independent optimal parameters. Both $K$ and $\sigma^2$ are hand-tuned

parameters. RBF features have been shown to enable algorithms to learn non-linear policies which greatly outperform their linear counterparts on non-linear tasks, such as walking [1], so we expected a performance increase. Now the task is to learn the optimal gain matrix $A$. As we don't have the labelled data to fit $A$, we need to use a reinforcement learning method.

### 7.2.3 Learning Policy Function

In order to learn the policy function $\mu(\boldsymbol{s})$ we use a contextual policy search algorithm called CREPS-CMA. CREPS-CMA is an extension of contextual REPS [17, 55] which is capable of multi-task learning. The goal of CREPS-CMA is to find a function $\mu(\boldsymbol{s})$ that given a context $\boldsymbol{s}$, it outputs a parameter vector $\boldsymbol{\theta}$ such that $\{\boldsymbol{s}, \boldsymbol{\theta}\}$ maximises the objective function $R(\boldsymbol{s}, \theta)$. The only accessible information on the objective function $R(\boldsymbol{s}, \theta)$ are evaluations $\{R_k\}_{k=1...k}$ of samples $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k\}_{k=1...k}$, where $k$ is the index of the sample, ranging from 1 to the number of samples $N$. CREPS-CMA maintains a stochastic search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ over the parameter space $\boldsymbol{\theta}$ of the objective function which is used to generate samples $\boldsymbol{\theta}$ given $\boldsymbol{s}$. The search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is modelled as a linear Gaussian policy, i.e.,

$$\pi(\boldsymbol{\theta}|s) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{A}^T\varphi(\boldsymbol{s}), \Sigma_\pi\right),$$

where the mean of the distribution is our policy function $\mu(\boldsymbol{s})$ we are searching for and covariance matrix $\Sigma_\pi$ controls the exploration of the algorithm. CREPS-CMA is an iterative algorithm. First it initializes the search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ by defining matrix and covariance matrix $\Sigma_\pi$ with arbitrary values[1]. Afterwards in each iteration, given context samples[2] $\{\boldsymbol{s}_k\}_{k=1...k}$, the current search distribution $q(\boldsymbol{\theta}|\boldsymbol{s})$ is used to create samples $\{\boldsymbol{\theta}_k\}_{k=1...k}$ of the parameter vector $\boldsymbol{\theta}$. Subsequently, the evaluation $\{R_k\}_{k=1...k}$ of samples $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k\}_{k=1...k}$ is obtained by querying the objective function $R(\boldsymbol{s}, \boldsymbol{\theta})$. And dataset $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, R_k\}_{k=1...k}$ is used to compute a weight $\{d_k\}_{k=1...k}$ for all samples. Each weight is a pseudo-probability for the corresponding sample. Subsequently, using $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, d_k\}_{k=1...k}$, a new Gaussian search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is estimated by estimating a new $A$ matrix and covariance matrix $\Sigma_\pi$. The new search distribution will give more probabilities to the samples $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k\}_{k=1...k}$ with better returns $\{R_k\}_{k=1...k}$. This process runs iteratively until the algorithm converges to a solution. After all we are interested in the matrix $A$ to construct our policy function

[1] With initializing we can define the region of the space that we would like the algorithm starts searching

[2] Please note that the way we sample contexts $\boldsymbol{s}_k$ depends on the task. Throughout this chapter we use a uniform distribution to sample contexts $\boldsymbol{s}_k$ which is desired kick distance. The intuition behind it is that all the kick distances have same importance for us.

$\mu(\boldsymbol{s})$. Algorithm 5 shows a compact representation of contextual stochastic search methods. Now we briefly explain how CREPS-CMA computes weights and what are

**Algorithm 5** Contextual stochastic search algorithm

**Initialize** $\pi(\theta|\boldsymbol{s})$
**Repeat**
**Set** $q(\theta|\boldsymbol{s})$ **to** $\pi(\theta|\boldsymbol{s})$
**Use a uniform distribution to generate context samples** $\{s_k\}_{k=1...N}$
**Sample parameters** $\{\theta_k\}_{k=1...N}$ **from current search distribution** $q(\theta|\boldsymbol{s})$ **given context samples** $\{s_k\}_{k=1...N}$
**Evaluate the reward** $R_k$ **of each sample in the sample set** $\{s_k, \theta_k\}_{k=1...N}$
**Use the data set** $\{\boldsymbol{\theta}_k, s_k, R_k\}_{k=1...N}$ **to compute a weight** $d_k$ **for each sample**
**Use the data set** $\{s_k, \theta_k, d_k\}_{k=1...N}$ **to update the new search distribution** $\pi(\theta|\boldsymbol{s})$
**Until search distribution** $\pi(\theta|\boldsymbol{s})$ **converges.**

the update rules of the search policy.

### 7.2.4 CREPS-CMA

The key idea behind contextual REPS [55] is to ensure a smooth and stable learning process by bounding the relative entropy between the old search distribution $q(\boldsymbol{\theta}|\boldsymbol{s})$ and the newly estimated policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ while maximising the expected return. This results in a weight

$$d_k = \exp\left((\mathcal{R}_{\boldsymbol{s\theta}} - V(\boldsymbol{s}))/\eta\right)$$

for each sample $[\boldsymbol{s}_k, \boldsymbol{\theta}_k]$, which we can use to estimate a new search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$. $\mathcal{R}_{\boldsymbol{s\theta}}$ denotes the expected performance when evaluating parameter vector $\boldsymbol{\theta}$ in context $\boldsymbol{s}$ and $V(\boldsymbol{s}) = \boldsymbol{\varphi}(\boldsymbol{s})^T\boldsymbol{w}$ is a context dependent baseline which is subtracted from the return $\mathcal{R}_{\boldsymbol{s\theta}}$. The parameters $\boldsymbol{w}$ and $\eta$ are Lagrangian multipliers that can be obtained by minimising the dual function, given as

$$\min_{\eta,\boldsymbol{w}} g(\eta, \boldsymbol{w}) = \eta\epsilon + \hat{\boldsymbol{\varphi}}^T\boldsymbol{w} + \eta \log\left(\sum_{K=1}^{N} \frac{1}{N} \exp\left(\frac{R^{[k]} - \boldsymbol{\varphi}(\boldsymbol{s}^{[k]})^T\boldsymbol{w}}{\eta}\right)\right).$$

Where $\hat{\boldsymbol{\varphi}} = \sum_{K=1}^{N} \boldsymbol{\varphi}(\boldsymbol{s}^{[k]})$ is the expected feature vector for the given context samples. We optimize this convex dual function by gradient decent. Now given dataset $\{\boldsymbol{s}_k, \boldsymbol{\theta}_k, d_k\}_{k=1...N}$ and the old Gaussian search distribution

$$q(\boldsymbol{\theta}|\boldsymbol{s}) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{A}_q^T\varphi(\boldsymbol{s}), \Sigma_q\right),$$

we want to find the new search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ by finding $A_\pi$ and $\Sigma_\pi$. Therefore we need two update rules, one for updating the context-dependent policy function $\boldsymbol{\mu}_\pi(\boldsymbol{s})$ of the search distribution and another one for updating the covariance matrix $\Sigma_\pi$ of the distribution.

### Context-Dependent Mean-Function Update Rule

The matrix $A$ can be obtained by the weighted maximum likelihood, i.e.,

$$\boldsymbol{A} = (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{U}, \tag{7.1}$$

where $\boldsymbol{\Phi}^T = [\varphi^{[1]}, ..., \varphi^{[N]}]$ contains the feature vector for all context samples $\{\boldsymbol{s}_k\}_{k=1...N}$, $\boldsymbol{U} = [\theta^{[1]}, ..., \theta^{[N]}]$ contains all the sample parameters, $\boldsymbol{D}$ is the diagonal weighting matrix containing the weightings $\{k\}_{k=1...N}$ and $\lambda \boldsymbol{I}$ is a regularization term. $\lambda$ is a very small number such as $1e-8$.

### Covariance Matrix Update Rule

Standard contextual REPS directly uses the weighted sample covariance matrix as $\boldsymbol{\Sigma}_\pi$ which is obtained by

$$\begin{aligned} \boldsymbol{S} &= \frac{\sum_{k=1}^{N} d_k \big(\boldsymbol{\theta}_k - \boldsymbol{A}^T \varphi(\boldsymbol{s}_k)\big)\big(\boldsymbol{\theta}_k - \boldsymbol{A}_\pi^T \varphi(\boldsymbol{s}_k)\big)^T}{Z}, \\ Z &= \frac{(\sum_{k=1}^{N} d_k)^2 - \sum_{k=1}^{N} (d_k)^2}{\sum_{k=1}^{N} (d_k)}. \end{aligned} \tag{7.2}$$

It has been shown that the sample covariance matrix from Equation 7.2 is not a good estimate of the true covariance matrix [5], since it biases the search distribution towards a specific region of the search space. In other words, the search distribution loses its exploration entropy along many dimensions of the parameter space, which causes premature convergence. This is a highly unwanted effect in policy search. To alleviate this problem, inspired by rank-$\mu$ update rule of CMA-ES [32], which is not a contextual algorithm, we combine the old covariance matrix and the sample covariance matrix from Equation 7.2, i.e.,

$$\boldsymbol{\Sigma}_\pi = (1-\lambda)\boldsymbol{\Sigma}_q + \lambda \boldsymbol{S}.$$

There are different ways to determine the interpolation factor $\lambda \in [0, 1]$ between the sample covariance matrix $\boldsymbol{S}$ and the old covariance matrix $\boldsymbol{\Sigma}_q$. For example, in [5], the factor $\lambda \in [0, 1]$ is chosen in such a way that the entropy of the new search distribution is reduced by a certain amount, while also being scaled with the number

of effective samples. We will extended REPS by using the rank-$\mu$ covariance matrix adaptation method of CMA-ES algorithm [32] which has been shown to be effective for avoiding premature convergence, i.e.,

$$\lambda = min\left(1, \frac{\phi_{\text{eff}}}{d_{\boldsymbol{\theta}}^2}\right), \phi_{\text{eff}} = \frac{1}{\sum_{k=1}^{N}(d^{[k]})^2}$$

where $\phi_{\text{eff}}$ is the number of effective samples and $d_{\boldsymbol{\theta}}$ is the dimension of the parameter space $\boldsymbol{\theta}$.

## 7.3 Experiments

[3] In this section, first we evaluate CREPS-CMA algorithm. Hence we use standard optimization test functions [67], such as the Sphere, the Rosenbrock and the Rastrigin (multi-modal) functions. We extend these functions to be applicable for the contextual setting. The task is to find the optimum 15 dimensional parameter vector $\boldsymbol{\theta}$ for a given 1 dimensional context $\boldsymbol{s}$. We will show that CREPS-CMA performs favourably. Afterwards, We use CREPS-CMA to optimize our kick controller for different desired kick distances for a simulated Nao robot[4] and will show our accuracy results, with both linear and non-linear policies. According to the results non-linear policy outperforms the linear one.

### 7.3.1 Standard Optimization Test Functions

We chose three standard optimization functions, which are the Sphere function

$$f(\boldsymbol{s}, \theta) = \sum_{i=1}^{p} \boldsymbol{x}_i^2,$$

the Rosenbrock function

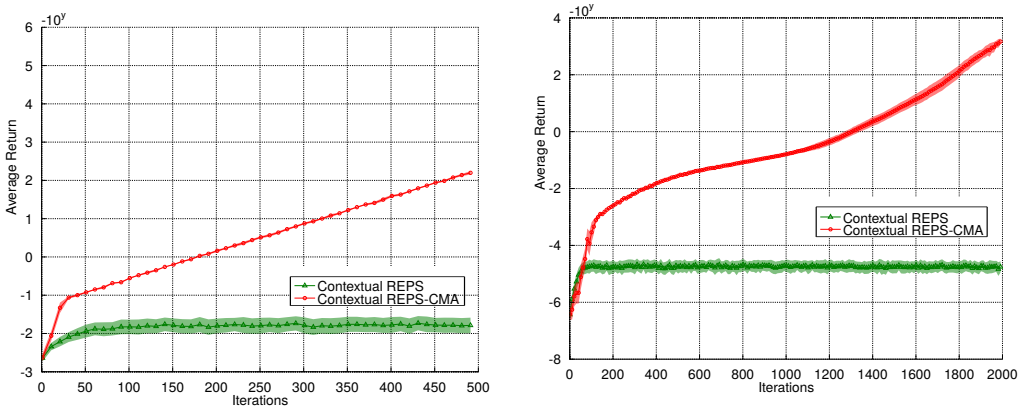
$$f(\boldsymbol{s}, \theta) = \sum_{i=1}^{p-1} [100(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i^2)^2 + (1 - \boldsymbol{x}_i)^2],$$

and also a multi-modal function, known as the Rastgirin function

$$f(\boldsymbol{s}, \theta) = 10p + \sum_{i=1}^{p} [\boldsymbol{x}_i^2 - 10\cos(2\pi \boldsymbol{x}_i)],$$

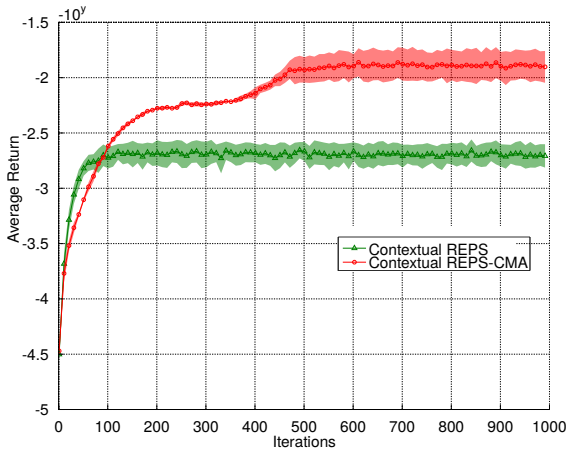
[3] Matlab source-code of CREPS-CMA algorithm is available on-line at https://dl.dropboxusercontent.com/u/16387578/ContextualREPSCMA.zip

[4] https://www.ald.softbankrobotics.com/en




(a) Sphere

(b) Rosenbrock




(c) Rastrigin

FIGURE 7.3: The performance comparison of CREPS and CREPS-CMA for optimising contextual versions of standard functions (a) Sphere, (b) Rosenbrock and (c) Rastrigin. The results show that CREPS-CMA clearly outperforms CREPS in all three benchmarks while CREPS suffers from premature convergence.

where $p$ is the number of dimensions of $\boldsymbol{\theta}$ and $\boldsymbol{x} = \boldsymbol{\theta} + \boldsymbol{A}\boldsymbol{s}$. The matrix $A$ is a constant matrix that was chosen randomly. In our case, because the context $\boldsymbol{s}$ is 1 dimensional, $A$ is a $p \times 1$ dimensional vector. Our definition for $\boldsymbol{x}$ means the optimum $\boldsymbol{\theta}$ for these functions is linearly dependent on the given context $\boldsymbol{s}$. The initial search area of $\boldsymbol{\theta}$ for all experiments is restricted to the hypercube $-5 \leq \boldsymbol{\theta}_i \leq 5, i = 1, \ldots, p$ and contexts are uniformly sampled from the interval $0 \leq \boldsymbol{s}_i \leq 3, i = 1, \ldots, z$ where $z$ is the dimension of the context space $\boldsymbol{s}$. In our experiments, the mean of the initial distributions has been chosen randomly in the defined search area. We compared CREPS-CMA with the standard Contextual REPS. In each iteration, we generated 50 new samples. The results in Figure 7.3 show that CREPS-CMA could successfully learn the contextual tasks while standard Contextual REPS suffers from premature convergence.

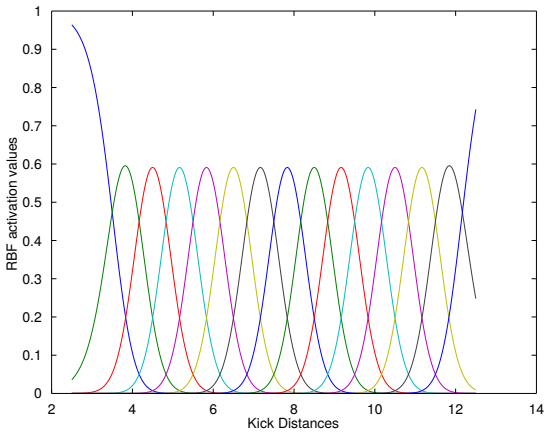



FIGURE 7.4: The 15 RBFs setup used for generating features.

### 7.3.2 Kick Task Results

We use a Nao humanoid robot simulated in RoboCup 3D simulation environment which is based on SimSpark [5]: a generic physical multiagent system simulator. The robot has 22 degrees of freedom, six in each leg, four in each arm, and two in the neck. We use CREPS-CMA to train a simulated NAO robot by optimising the kick controller explained in section 2 using both linear policies, i.e., $\varphi(\boldsymbol{s}^{[i]}) = [1 \quad \boldsymbol{s}^{[i]}]$ and a RBF based non-linear policy. The desired kick distance $s$ varies from $2.5$m to $12.5$m. For the non-linear policy, we choose $K = 15$ normalized RBFs and $\sigma^2$ is set to 0.5. Both $K$ and the $\sigma^2$ parameters were chosen by trial and error to maximize the results accuracy. Figure 7.4 shows the setup of the used RBFs over the context range.

We maximize a context dependent objective function

$$R(s, \theta) = -(x - s)^2 - y^2,$$

where $s$ is the desired kick distance, and $x$ and $y$ are the ball distances travelled along the $x$- and $y$-axes using the kick controller with the given parameter set $\theta$. We initialize the search distribution $\pi$ with a hand tuned kick policy, which was able to kick the ball over $15m$. We optimized the kick with $1000$ iterations. Each iteration generates $20$ new samples where the contexts were sampled uniformly. Each sample was evaluated $5$ times, and was averaged to smooth out the noisy returns. In order to simulate competition conditions, for evaluating each sample, we placed the robot in 5 different positions around the ball and it had to perceive the ball, move towards it, position itself in place and then kick it towards the target goal using the kick controller. We compared the performance of the linear policy with non-linear one. Figure 7.6 shows that the non-linear policy clearly outperforms the linear one and the

[5] http://simspark.sourceforge.net/

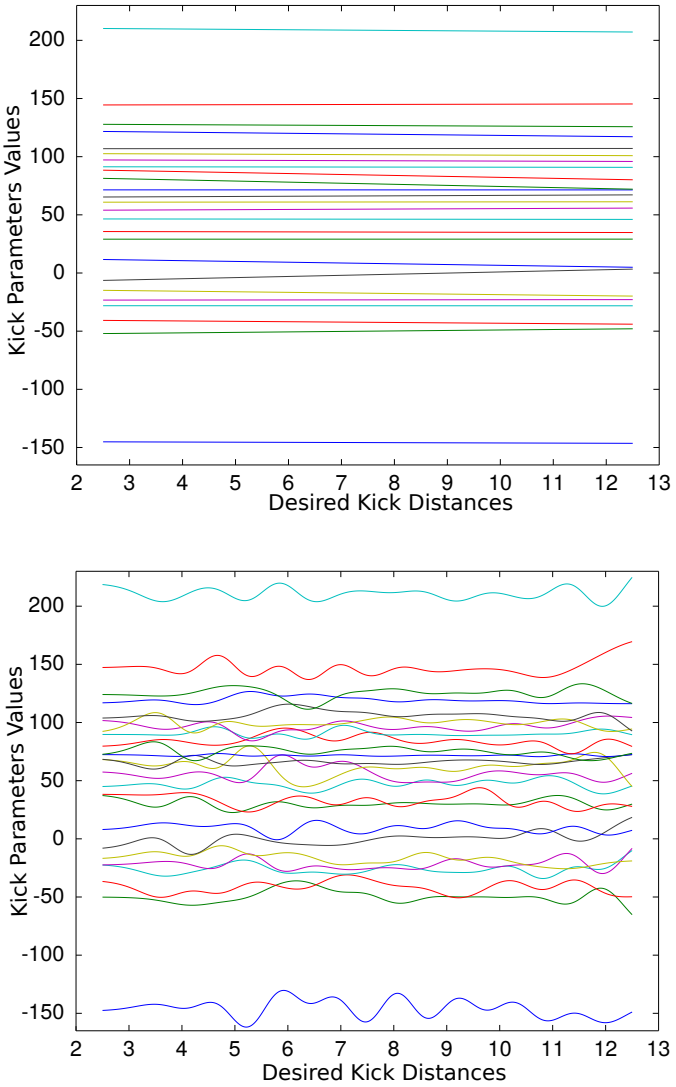



FIGURE 7.5: The learned linear (left) and non-linear (right) policies for kick distances of 2.5 to 12.5 meters. The y-axis represents the controller parameter values for a given desired kick distance, and the x-axis represents the desired kick distance.

accuracy of the non-linear policy is considerable. [6] The average error of the linear policy was $0.82 \pm 0.10m$ while we achieved an average error of $0.34 \pm 0.11m$ using the non-linear policy. As expected, using a non-linear policy improves the accuracy of the results with order of magnitude. In fact, the average error is more than halved. This also demonstrates the non-linearity nature of robotic tasks such as kick task and the usefulness of using RBF functions to capture this non-linearity. Figure 7.5 shows the learned linear and non-linear policies for generalizing the 25 parameter

[6] Demonstration video of the non-linear kick controller using the magma challenge tool[**MCT2015**] is available on-line at https://www.dropbox.com/sh/0iimyykf6xejj6g/AADg9iCNJZAbu3Voe2UKsmQza?dl=0.

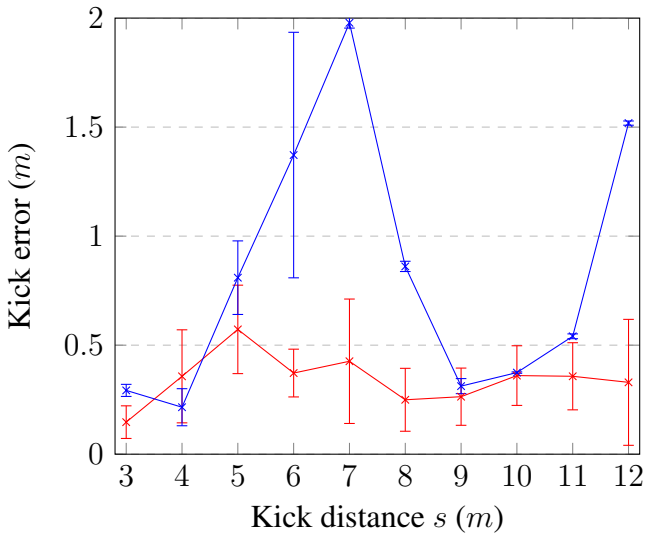



FIGURE 7.6: The performance of the learned linear (blue) and non-linear (red) policies. The x-axis represents the desired kick distance, in meters, while the y-axis represents the error with respect to desired kick distance, also in meters.

kick controller for different kick distances.We can see that the learned linear policy is a linear approximation of its corresponding non-linear policy.

## 7.4 Conclusion

We used a recently proposed contextual policy search algorithm to generalize a robot kick controller for different desired kick distances, where a context is described by a real-valued vector of distances. We have modified the algorithm, naming it CREPS-CMA. Using CREPS-CMA, we have successfully learned linear and non-linear policies over the context of kick distances. The non-linear policy outperforms its linear counterpart, and allows a humanoid robot to kick a ball with flexible distances and with satisfactory accuracy results, which could leads to a better control and coordination in a robotic soccer match. In this research, we also demonstrated the non-linearity of a kick task. In future we will use more complex kick controllers such as dynamic motor primitives.

# Chapter 8

# Summary and future work

In chapter one we gave a foundation for entire thesis on continuous back box optimisation.

In chapter 2, we compared different methods for estimating the covariance matrix of a Gaussian policy for weighted maximum likelihood estimate(MLE) based policy search methods. Weighted ML estimate of covariance matrices is an unreliable estimator with a high variance. The use of WMLE leads to over-fitted covariance estimates, and, hence the variance/entropy of the policy decreases too quickly, which may cause premature convergence. We proposed a new algorithm called Covariance Estimation with Controlled the Entropy Reduction(CECER). We showed that using the CECER, we could control the entropy reduction of the policy and get a better covariance matrix approximation, which results in an significant improved performance of the policy search algorithm.

In chapter 3 we derived the full CMA-ES update equations for mean and covariance with an expectation-maximization based framework using information-geometric trust regions. The presented update for the covariance matrix share the same structure then the CMA-ES algorithm. However, CMA-ES is not using a trust region (i.e., use constraint on KL), but a penalty on KL is used in the objective as regularizer. As a consequence, the optimum 'Lagrangian' multipliers in CMA-ES are set by hand and remain fixed during the learning. However they are well established and can be left constant throughout many applications. In the trust region formulation, the Lagrangian multipliers are optimized for the given bound $\epsilon$. As we show, in addition to the theoretical foundation, this optimisation of the parameters gives us an improved performance over the original algorithm. Moreover, we also use the KL bound to obtain update rules for the mean and the step size parameters. CMA-ES does not use a regularizer for the mean update but directly use the unregularized maximum likelihood update. While the mean does not easily overfit for low-dimensional parameters spaces with a high number of individuals, an unregularized update is more problematic for high-dimensional parameter spaces where it might even diverge. In contrast to the mean and the covariance update, our step-size update does not match

the step-size update from CMA-ES. Both, the TR-CMA-ES and CMA-ES take advantage of evolution path to set step size $\sigma$. However our update rule for the step size is obtained from the same principle as used for the mean and the covariance matrix. Given the similarities in terms of the mean and covariance matrix update between CMA-ES and our algorithm, our step size control update rule is more consistent and a more principled than the update rule is used in standard CMA-ES. The new step-size update also performs favourably in our experiments and should also be preferred for CMA-ES due to the consistent derivation. Our algorithm also enjoys all the invariance properties of the CMA-ES.

In chapter 4 we investigated contextual stochastic search methods for multi task learning. Stochastic search methods such as CMA-ES have been employed extensively for black box optimization. However, these algorithms lack the important feature of contextual learning. Therefore we extended CMA-ES for contextual setting while we also provide a new theoretical justification for its covariance update rule. It turns out using baseline, the old covariance matrix and the step size control are crucial ingredients for a competitive performance. One interesting observation is that contextual learning also facilitates learning single tasks. The reason is that easier tasks can guide the optimisation for learning harder tasks.

In chapter 5 we introduced MORE, a stochastic search algorithm that use quadratic models for its updates. Using KL-bounds to limit the update of the search distribution is a wide-spread idea in the stochastic search community but typically requires approximations. However MORE satisfy the KL-bound analytically. By relying on a Gaussian search distribution and on locally learned quadratic models of the objective function, we could obtain a closed form of the information theoretic policy update. We also introduced an additional entropy term in the formulation that is needed to avoid premature shrinkage of the variance of the search distribution. Our algorithm considerably outperforms competing methods in all the considered scenarios. The main disadvantage of MORE is the number of parameters. However based on our experiments, these parameters are not problem specific. in chapter 5, we investigated the Contextual stochastic search algorithms. However the search distribution typically uses a parametric model that is linear in the some hand-defined context features. Finding good context features is a challenging task, and hence, non-parametric methods are often preferred over their parametric counter-parts. Therefore, in chapter 6, we propose a non-parametric contextual stochastic search algorithm that can learn a non-parametric search distribution for multiple tasks simultaneously. In difference to existing methods, our method can also learn a context dependent covariance matrix that guides the exploration of the search process. We illustrate its performance on several non-linear contextual tasks.

In chapter 7, as a practical robotics application for contextual policy search, we studied generalizing a kick controller for different contexts. In order to do so we used contextual Relative Entropy Policy Search. Using CREPS, we successfully learned a policy that generalizes the kick skill for different distances.

For future work, we will investigate other theoretical aspects of the presented algorithms such as the theoretical difference between the forward KL and inverse KL trust region. We will also extend our framework for full reinforcement learning problem. We will investigate the methods to set the bandwidth of the kernel function of non parametric contextual stochastic search algorithm automatically. And we will also investigate using richer function approximators such as neural networks as a representation for the contextual functions. Also learning different solutions when the function is multi modal is another interesting future direction. Finally we will investigate the applicability of our algorithms on real robots.

# Bibliography


[1] A. Abdolmaleki et al. “Contextual Policy Search for Generalizing a Parameterized Biped Walking Controller”. In: *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2015.

[2] A. Abdolmaleki et al. “Contextual Policy Search for Linear and Nonlinear Generalization of a Humanoid Walking Controller”. In: *Journal of Intelligent and Robotic Systems*. 2016.

[3] A. Abdolmaleki et al. “Learning a Humanoid Kick With Controlled Distance”. In: *Robocup Symposium*. 2016.

[4] A. Abdolmaleki et al. “Model Based Relative Entropy Stochastic Search”. In: *NIPS*. 2015.

[5] A. Abdolmaleki et al. “Regularized covariance estimation for weighted maximum likelihood policy search methods”. In: *Humanoids Conference*. 2015.

[6] Y. Akimoto et al. “Theoretical foundation for CMA-ES from information geometry perspective”. In: *Algorithmica* (2012).

[7] S. Amari. “Natural Gradient Works Efficiently in Learning”. In: *Neural Computation* (2 1998). ISSN: 0899-7667. DOI: 10.1162/089976698300017746. URL: http://dl.acm.org/citation.cfm?id=287476.287477.

[8] P Bayer, CM Bürger, and M Finkel. “Solving computationally-demanding reliability-based design problems in hydrogeology”. In: *IAHS-AISH publication* (2008), pp. 22–26.

[9] Hans-Georg Beyer and Hans-Paul Schwefel. “Evolution Strategies &Ndash;A Comprehensive Introduction”. In: (2002).

[10] Andrew J Booker et al. “Optimization using surrogate objectives on a helicopter test example”. In: *Computational Methods for Optimal Design and Control*. Springer, 1998, pp. 49–58.

[11] G.E.P. Box and K.G. Wilson. “On the Experimental Attainment of Optimum Conditions”. In: (1951).

[12] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[13] Roberto Calandra et al. "An experimental comparison of Bayesian optimization for bipedal locomotion". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 1951–1958.

[14] Rich Caruana. "Multitask Learning". PhD thesis. CMU, 1997.

[15] Christophe Charbuillet et al. "Optimizing feature complementarity by evolution strategy: Application to automatic speaker verification". In: *Speech Communication* 51.9 (2009), pp. 724–731.

[16] Bruno Da Silva, George Konidaris, and Andrew Barto. "Learning Parameterized Skills". In: *International Conference on Machine Learning (ICML)* (2012).

[17] C. Daniel, G. Neumann, and J. Peters. "Hierarchical Relative Entropy Policy Search". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2012.

[18] Charles Darwin. *On the Origin of Species by Means of Natural Selection*. Murray, 1859.

[19] Peter Dayan and Geoffrey E. Hinton. "Using expectation-maximization for reinforcement learning". In: *Neural Comput.* 9.2 (1997), pp. 271–278. ISSN: 0899-7667. DOI: http://dx.doi.org/10.1162/neco.1997.9.2.271.

[20] M. P. Deisenroth, G. Neumann, and J. Peters. "A Survey on Policy Search for Robotics". In: (2013). URL: http://www.ias.tu-darmstadt.de/uploads/Site/EditPublication/PolicySearchReview.pdf.

[21] Marc Peter Deisenroth et al. "Multi-task policy search for robotics". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014.

[22] A. Dempster, N. Laird, and D. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society, Series B* 39.1 (1977), pp. 1–38.

[23] Mike Depinet, Patrick MacAlpine, and Peter Stone. "Keyframe sampling, optimization, and behavior integration: Towards long-distance kicking in the robocup 3d simulation league". In: *RoboCup 2014: Robot World Cup XVIII*. Springer, 2014, pp. 571–582.

[24] Rui Ferreira et al. “Development of an omnidirectional kick for a nao humanoid robot”. In: *Advances in Artificial Intelligence–IBERAMIA 2012*. Springer, 2012, pp. 571–580.

[25] Roger Fletcher. *Practical Methods of Optimization*. Second. John Wiley & Sons, 1987.

[26] Frauke Friedrichs and Christian Igel. “Evolutionary Tuning of Multiple SVM Parameters”. In: *Neurocomputing* (2005).

[27] T. Furmston and D. Barber. “A Unifying Perspective of Parametric Policy Search Methods for Markov Decision Processes”. In: *Neural Information Processing Systems (NIPS)*. 2012.

[28] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[29] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. “Accelerated neural evolution through cooperatively coevolved synapses”. In: *Journal of Machine Learning Research* 9.May (2008), pp. 937–965.

[30] S. Ha and C.K Liu. “Evolutionary Optimization for Parameterized Whole-body Dynamic Motor Skills”. In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. 2016.

[31] N. Hansen. *The CMA Evolution Strategy: A Tutorial*. Tutorial. 2016.

[32] N. Hansen, S.D. Muller, and P. Koumoutsakos. “Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES).” In: *Evolutionary Computation* (2003).

[33] Nikolaus Hansen and Andreas Ostermeier. “Completely Derandomized Self-Adaptation in Evolution Strategies”. In: *Evolutionary Computation* (2001).

[34] Nikolaus Hansen et al. “A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion”. In: *IEEE Transactions on Evolutionary Computation* 13.1 (2009), pp. 180–197.

[35] Martina Hasenjäger et al. “Three dimensional evolutionary aerodynamic design optimization with CMA-ES”. In: *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM. 2005, pp. 2173–2180.

[36] W. K. Hastings. “Monte Carlo sampling methods using Markov chains and their applications”. In: *Biometrika* (1970).

[37] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[38] Jemin Hwangbo et al. “ROCK—Efficient black-box optimization for policy learning”. In: *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE. 2014, pp. 535–540.

[39] Oscar Ibáñez et al. “An experimental study on the applicability of evolutionary algorithms to craniofacial superimposition in forensic identification”. In: *Information Sciences* 179.23 (2009), pp. 3998–4028.

[40] C. Igel, T. Suttorp, and N. Hansen. “ A computational efficient covariance matrix update and a (1+ 1)-CMA for evolution strategies.” In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006.

[41] A. Ijspeert and S. Schaal. “Learning Attractor Landscapes for Learning Motor Primitives”. In: *Advances in Neural Information Processing Systems 15(NIPS)*. 2003.

[42] Mohamed Jebalia et al. “Identification of the isotherm function in chromatography using CMA-ES”. In: *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE. 2007, pp. 4289–4296.

[43] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[44] Donald R Jones. “A taxonomy of global optimization methods based on response surfaces”. In: *Journal of global optimization* 21.4 (2001), pp. 345–383.

[45] Donald R Jones, Matthias Schonlau, and William J Welch. “Efficient global optimization of expensive black-box functions”. In: *Journal of Global optimization* 13.4 (1998), pp. 455–492.

[46] Jérôme Henri Kämpf and Darren Robinson. “A hybrid CMA-ES and HDE optimisation algorithm with application to solar energy potential”. In: *Applied Soft Computing* 9.2 (2009), pp. 738–745.

[47] H. Karshenas et al. “Regularized continuous estimation of distribution algorithms.” In: *Applied Soft Computing* (2013).

[48] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers Inc., 2001.

[49] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by simulated annealing”. In: *SCIENCE* (1983).

[50] Jurgen Klockgether and Hans-Paul Schwefel. “Two-phase nozzle and hollow core jet experiments”. In: *Proc. 11th Symp. Engineering Aspects of Magnetohydrodynamics*. Pasadena, CA: California Institute of Technology. 1970, pp. 141–148.

[51] J. Kober, E. Oztop, and J. Peters. “Reinforcement Learning to adjust Robot Movements to New Situations”. In: *Proceedings of the Robotics: Science and Systems Conference (RSS)*. 2010.

[52] J. Kober and J. Peters. “Policy search for motor primitives in robotics”. In: *Neural Information Processing Systems (NIPS)*. 2009.

[53] J. Kober and J. Peters. “Policy Search for Motor Primitives in Robotics”. In: *Machine Learning* (2010), pp. 1–33.

[54] Jens Kober, Betty J. Mohler, and Jan Peters. “Learning Perceptual Coupling for Motor Primitives”. In: *Intelligent Robots and Systems (IROS)*. 2008, pp. 834–839.

[55] A. Kupcsik et al. “Data-Efficient Contextual Policy Search for Robot Movement Skills”. In: *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. 2013.

[56] Pedro Larraanaga and Jose A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.

[57] J Leitner, C Ampatzis, and D Izzo. “Evolving ANNs for spacecraft rendezvous and docking”. In: *10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*. 2010, p. 8.

[58] I. Loshchilov, M. Schoenauer, and M. Sebag. “Intensive Surrogate Model Exploitation in Self-Adaptive Surrogate-Assisted CMA-ES (SAACM-ES)”. In: *GECCO*. 2013.

[59] I. Loshchilov, M. Schoenauer, and M. Sebag. “KL-based Control of the Learning Schedule for Surrogate Black-Box Optimization”. In: *CoRR* (2013).

[60] Patrick MacAlpine, Mike Depinet, and Peter Stone. “UT Austin Villa 2014: RoboCup 3D Simulation League Champion via Overlapping Layered Learning”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*. 2015.

[61] S. Mannor, R. Rubinstein, and Y. Gat. “The Cross Entropy method for Fast Policy Search”. In: *Proceedings of the 20th International Conference on Machine Learning(ICML)*. 2003.

[62] G. Mehmet. “Bayesian Supervised Dimensionality Reduction”. In: *IEEE T. Cybernetics* (2013).

[63] Silvio Priem Mendes et al. “A differential evolution based algorithm to optimize the radio network design problem”. In: *e-Science and Grid Computing, 2006. e-Science'06. Second IEEE International Conference on*. IEEE. 2006, pp. 119–119.

[64] N. Metropolis et al. “Equation of State Calculations by Fast Computing Machines”. In: *The Journal of Chemical Physics* (1953).

[65] E Miguez, E Diaz-Dorado, and J Cidras. “An application of an evolution strategy in power distribution system planning”. In: *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE. 1998, pp. 241–246.

[66] EA Minisci and Giulio Avanzini. “Comparative study on the application of evolutionary optimization techniques to orbit transfer maneuvers”. In: (2008).

[67] M. Molga and C. Smutnicki. “Test Functions for Optimization Needs”. In: *http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf*. 2005.

[68] Andrew W. Moore and Jeff G. Schneider. “Memory-based Stochastic Optimization”. In: *Advances in Neural Information Processing Systems 8, NIPS*. 1995.

[69] I. Murray, R.P. Adams, and D.J.C. MacKay. “Elliptical Slice Sampling”. In: *JMLR: W&CP* 9 (2010).

[70] Sibylle D. Müller et al. “Optimization Based on Bacterial Chemotaxis”. In: *IEEE Transactions on Evolutionary Computation* (2002).

[71] J. A. Nelder and R. Mead. “A Simplex Method for Function Minimization”. In: *Computer Journal* (1965).

[72] G. Neumann. “Variational Inference for Policy Search in Changing Situations”. In: *ICML*. Bellevue, Washington, USA, 2011.

[73] Cord Niehaus, Thomas Röfer, and Tim Laue. “Gait optimization on a humanoid robot using particle swarm optimization”. In: *Proceedings of the Second Workshop on Humanoid Soccer Robots in conjunction with the*. 2007, pp. 1–7.

[74] Y. Ollivier et al. “ Information-geometric optimization algorithms: A unifying picture via invariance principles.” In: *arXiv preprint arXiv:1106.3708*. 2011.

[75] J. Peters, K. Mülling, and Y. Altun. "Relative Entropy Policy Search". In: *AAAI*. AAAI Press, 2010.

[76] J. Peters and S. Schaal. "Natural Actor-Critic". In: *Neurocomputation* 71.7-9 (2008), pp. 1180–1190. ISSN: 0925-2312.

[77] M.J.D. Powell. "The BOBYQA Algorithm for Bound Constrained Optimization Without Derivatives". In: *Report DAMTP 2009/NA06, University of Cambridge*. 2009.

[78] M.J.D. Powell. "The NEWUOA Software for Unconstrained Optimization without Derivatives". In: *Report DAMTP 2004/NA05, University of Cambridge*. 2004.

[79] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[80] I. Rechenberg. *Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Friedrich Frommann Verlag, 1973.

[81] R. Ros and N. Hansen. "A simple modification in CMA-ES achieving linear time and space complexity". In: *Parallel Problem Solving from Nature*. 2008, pp. 296–305.

[82] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2004.

[83] T. Rückstieß, M. Felder, and J. Schmidhuber. "State-Dependent Exploration for Policy Gradient Methods". In: *Proceedings of the European Conference on Machine Learning (ECML)*. 2008.

[84] Pole In C Ryan et al. "References to CMA-ES Applications". In: *Strategies* 4527.467 (2007).

[85] Tim Salimans et al. "Evolution strategies as a scalable alternative to reinforcement learning". In: *arXiv preprint arXiv:1703.03864* (2017).

[86] J. Schulman et al. "Trust Region Policy Optimization". In: *ICML*. 2015.

[87] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie - Mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie*. Birkhäuser, 1977.

[88] J. Schäfer and K. Strimmer. "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics." In: *Statistical applications in genetics and molecular biology* (2005).

[89] Rainer Storn and Kenneth Price. *Differential Evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. International Computer Science Institute, Berkeley*. Tech. rep. CA, 1995, Tech. Rep. TR-95–012, 1995.

[90] F. Stulp and O. Sigaud. "Path Integral Policy Improvement with Covariance Matrix Adaptation". In: *International Conference on Machine Learning (ICML)*. 2012.

[91] Freek Stulp et al. "Learning compact parameterized skills with a single regression". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2013.

[92] Y. Sun et al. "Efficient Natural Evolution Strategies". In: *GECCO)*. Montreal, Canada, 2009. ISBN: 978-1-60558-325-9. DOI: http://doi.acm.org/10.1145/1569901.1569976. URL: http://doi.acm.org/10.1145/1569901.1569976.

[93] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. Boston, MA: MIT Press, 1998.

[94] E. Theodorou, J. Buchli, and S. Schaal. "A Generalized Path Integral Control Approach to Reinforcement Learning". In: *The Journal of Machine Learning Research* (2010).

[95] David J Wales and Jonathan PK Doye. "Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms". In: *The Journal of Physical Chemistry A* 101.28 (1997), pp. 5111–5116.

[96] Jack M Wang, David J Fleet, and Aaron Hertzmann. "Optimizing walking controllers". In: *ACM Transactions on Graphics (TOG)* 28.5 (2009), p. 168.

[97] S-K Wang, J-P Chiou, and C-W Liu. "Non-smooth/non-convex economic dispatch by a novel hybrid differential evolution algorithm". In: *IET Generation, Transmission & Distribution* 1.5 (2007), pp. 793–803.

[98] D. Wierstra et al. "Natural Evolution Strategies". In: *Journal of Machine Learning Research* (2014).

[99] Daan Wierstra et al. "Fitness expectation maximization". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2008, pp. 337–346.

[100] Susanne Winter, Bernhard Brendel, and Christian Igel. “Registration of bone structures in 3D ultrasound and CT data: Comparison of different optimization strategies”. In: *International Congress Series*. Vol. 1281. Elsevier. 2005, pp. 242–247.